

Open Client/Server™ Configuration Guide for UNIX

Open Client/Server Release 11.1.x

Document ID: 35831-01-1110-03

Last Revised: July 01, 1997

Principal author: Connectivity Products Group

Document ID: 35831-01-1110

This publication pertains to Open Client/Server Release 11.1.x of the Sybase database management software and to any subsequent release until otherwise indicated in new editions or technical notes. Information in this document is subject to change without notice. The software described herein is furnished under a license agreement, and it may be used or copied only in accordance with the terms of that agreement.

Document Orders

To order additional documents, U.S. and Canadian customers should call Customer Fulfillment at (800) 685-8225, fax (617) 229-9845.

Customers in other countries with a U.S. license agreement may contact Customer Fulfillment via the above fax number. All other international customers should contact their Sybase subsidiary or local distributor.

Upgrades are provided only at regularly scheduled software release dates.

Copyright © 1989–1997 by Sybase, Inc. All rights reserved.

No part of this publication may be reproduced, transmitted, or translated in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without the prior written permission of Sybase, Inc.

Sybase Trademarks

Sybase, the Sybase logo, APT-FORMS, Certified SYBASE Professional, Data Workbench, Deft, First Impression, GainExposure, Gain *Momentum*, PowerBuilder, Powersoft, Replication Server, S-Designor, SQL Advantage, SQL Debug, SQL SMART, SQL Solutions, Transact-SQL, VisualWriter, and VQL are registered trademarks of Sybase, Inc. Adaptable Windowing Environment, Adaptive Server, ADA Workbench, AnswerBase, Application Manager, AppModeler, APT-Build, APT-Edit, APT-Execute, APT-Library, APT-Translator, APT Workbench, Backup Server, BayCam, Bit-Wise, Client-Library, Client/Server Architecture for the Online Enterprise, Client/Server for the Real World, Client Services, CodeBank, Column Design, Connection Manager, DataArchitect, Database Analyzer, DataExpress, Data Pipeline, DataWindow, DB-Library, Deft Analyst, Deft Designer, Deft Educational, Deft Professional, Deft Trial, Designor, Developers Workbench, Dimensions Anywhere, Dimensions Enterprise, Dimensions Server, DirectCONNECT, Easy SQR, Embedded SQL, EMS, Enterprise Builder, Enterprise Client/Server, Enterprise CONNECT, Enterprise Manager, Enterprise SQL Server Manager, Enterprise Work Architecture, Enterprise Work Designer, Enterprise Work Modeler, EWA, Formula One, Gateway Manager, GeoPoint, InfoMaker, InformationCONNECT, Intermedia Server, InternetBuilder, iScript, KnowledgeBase, MainframeCONNECT, Maintenance Express, MAP, MDI, MDI Access Server, MDI Database Gateway, media.splash, MetaWorks, MethodSet, Movedb, Navigation Server Manager, Net-Gateway, NetImpact, Net-Library, New Media Studio, ObjectCONNECT, ObjectCycle, OmniCONNECT, OmniSQL

Access Module, OmniSQL Server, OmniSQL Toolkit, Open Client, Open ClientCONNECT, Open Client/Server, Open Client/Server Interfaces, Open Gateway, Open Server, Open ServerCONNECT, Open Solutions, Optima++, PB-Gen, PC APT-Execute, PC DB-Net, PC Net Library, PowerBuilt, PowerBuilt with PowerBuilder, PowerScript, PowerSocket, Powersoft Portfolio, Power Through Knowledge, PowerWare Desktop, PowerWare Enterprise, ProcessAnalyst, Replication Agent, Replication Driver, Replication Server Manager, Report-Execute, Report Workbench, Resource Manager, RW-DisplayLib, RW-Library, SAFE, SDF, Secure SQL Server, Secure SQL Toolset, Security Guardian, SKILS, smart.partners, smart.parts, smart.script, SQL Anywhere, SQL Central, SQL Code Checker, SQL Edit, SQL Edit/TPU, SQL Remote, SQL Server, SQL Server/CFT, SQL Server/DBM, SQL Server Manager, SQL Server Monitor, SQL Server SNMP SubAgent, SQL Station, SQL Toolset, StarDesigner, Sybase Client/Server Interfaces, Sybase Development Framework, Sybase Dimensions, Sybase Gateways, Sybase Intermedia, Sybase Interplay, Sybase IQ, Sybase MPP, Sybase SQL Desktop, Sybase SQL Lifecycle, Sybase SQL Workgroup, Sybase Synergy Program, Sybase Virtual Server Architecture, Sybase User Workbench, SybaseWare, SyBooks, System 10, System 11, the System XI logo, SystemTools, Tabular Data Stream, The Architecture for Change, The Enterprise Client/Server Company, The Online Information Center, Turning Imagination Into Reality, Visual Components, VisualSpeller, Warehouse WORKS, Watcom, Watcom SQL, Watcom SQL Server, web.sql, WebSights, WebViewer, WorkGroup SQL Server, XA-Library, and XA-Server are trademarks of Sybase, Inc. 1/97

All other company and product names used herein may be trademarks or registered trademarks of their respective companies.

Restricted Rights

Use, duplication, or disclosure by the government is subject to the restrictions set forth in subparagraph (c)(1)(ii) of DFARS 52.227-7013 for the DOD and as set forth in FAR 52.227-19(a)-(d) for civilian agencies.

Sybase, Inc., 6475 Christie Avenue, Emeryville, CA 94608.

Table of Contents

About This Book

Audience	xiii
How to Use This Book	xiii
Configuration Instructions	xiii
Configuration Utilities	xiv
Configuration Reference	xiv
Related Documents	xv
Other Sources of Information	xvi
Electronic Information Sources	xvi
Support <i>Plus</i> Online Services	xvi
SyBooks-on-the-Web	xvii
Conventions	xvii
If You Need Help	xviii

Part 1: Configuration Instructions

1. Configuration Overview

About Open Client and Open Server	1-1
Overview of Configuration	1-1
The Initialization Process	1-2
The Connection Process	1-2
Configuration Tasks	1-3

2. Basic Configuration for Open Client

Overview of Basic Configuration	2-1
Configuration Tasks	2-3

3. Basic Configuration for Open Server

About Open Server Applications	3-1
Overview of Basic Configuration	3-1
Configuration Tasks	3-3

4. Using a Directory Service

Overview of Directory Services	4-1
Server Objects and Attributes	4-1
How Applications Use a Directory Service	4-3
Configuration Tasks	4-5

5. Using Security Services

Overview of Network-Based Security	5-1
Security Mechanisms	5-1
Security Drivers	5-2
Security Services	5-2
How Applications Use Security Services	5-3
Client-Library and Security Services	5-4
Server Library and Security Services	5-5
Configuration Tasks	5-6

Part 2: Configuration Utilities

6. Using *dscp* and *dscp_dce*

About <i>dscp</i> and <i>dscp_dce</i>	6-1
Starting <i>dscp</i> and <i>dscp_dce</i>	6-3
Viewing Your Configuration	6-3
Getting Help	6-3
Using <i>dscp</i> and <i>dscp_dce</i> Sessions	6-4
Opening a Session	6-4
Listing Sessions	6-4
Switching Between Open Sessions	6-4
Closing a Session	6-5
Adding and Modifying Server Entries	6-5
Listing Server Entries	6-7
Viewing a Server Entry	6-7
Adding a Server Entry	6-8
Modifying a Server Entry	6-8
Deleting Entries	6-9
Copying Server Entries	6-10
Copying Entries Within a Session	6-10
Copying Entries Between Sessions	6-10
Copying All Entries to a Different Session	6-11

Exiting <i>dscp</i> and <i>dscp_dce</i>	6-12
---	------

7. Using *dsedit* and *dsedit_dce*

About <i>dsedit</i> and <i>dsedit_dce</i>	7-1
Starting <i>dsedit_dce</i> and <i>dsedit_dce</i>	7-1
Getting Help	7-2
Opening a Session	7-2
Interfaces file Sessions	7-2
DCE Directory Sessions	7-4
Adding, Viewing, and Editing Server Entries	7-6
Adding or Editing Network Transport Addresses	7-7
TCP/IP Addresses	7-7
SPX/IPX Addresses	7-7
Troubleshooting <i>dsedit</i> or <i>dsedit_dce</i> Problems	7-8
<i>dsedit</i> or <i>dsedit_dce</i> Does Not Start	7-8
DCE Does Not Display As an Available Directory Service	7-8
Cannot Open DCE Directory Session	7-9
Cannot Add, Modify, or Delete Server Entries	7-9

Part 3: Configuration Reference

A. Environment Variables

Environment Variables Used for Connection	A-1
Environment Variables Used for Localization	A-1
Setting Environment Variables	A-2
C Shell	A-2
Bourne Shell	A-2

B. Configuration Files

About Configuration Files	B-1
<i>libtcl.cfg</i>	B-1
Dynamic Linking of Drivers	B-2
How <i>libtcl.cfg</i> Is Used	B-2
Location of <i>libtcl.cfg</i>	B-3
Layout of <i>libtcl.cfg</i>	B-3
DIRECTORY Section	B-3
SECURITY Section	B-4
DRIVERS Section	B-6

Network Drivers	B-7
Adding a Directory Driver	B-7
Adding a Security Driver	B-8
Adding a Network Driver	B-9
<i>interfaces</i>	B-10
How It Is Used	B-11
Location of <i>interfaces</i>	B-11
<i>interfaces</i> Entries	B-11
Standard Format	B-11
Transport Layer Interface Format	B-14
Editing <i>interfaces</i>	B-15
Standby Server Addressing	B-15
<i>ocs.cfg</i> File	B-16

C. Localization

Overview of the Localization Process	C-1
Environment Variables Used During Localization	C-1
Localization Files	C-3
The <i>locales</i> Directory	C-5
<i>locales.dat</i>	C-5
How It Is Used	C-5
Location of <i>locales.dat</i>	C-5
<i>locales.dat</i> Sections and Entries	C-5
<i>locales.dat</i> File Fragment	C-6
Editing <i>locales.dat</i>	C-6
Localized Message Files	C-8
The <i>charsets</i> Directory	C-8
Conversion Configuration Files	C-9
How Conversion Configuration Files Are Used	C-9
Location of the Conversion Configuration Files	C-10
Conversion Configuration File Entries	C-10
Conversion Configuration File Example	C-11
Collating Sequence Files	C-11
The <i>config</i> Directory	C-12
<i>objectid.dat</i>	C-12
Location of <i>objectid.dat</i>	C-12
<i>objectid.dat</i> Sections and Entries	C-12
<i>objectid.dat</i> File Fragment	C-12
Editing <i>objectid.dat</i>	C-13
<i>mnemonic.dat</i>	C-13

How <i>mnemonic.dat</i> Is Used	C-13
Location of <i>mnemonic.dat</i>	C-14
<i>mnemonic.dat</i> Entries	C-14
<i>mnemonic.dat</i> Example	C-14
Adding Strings to <i>mnemonic.dat</i>	C-15

D. DCE Security Services

Supported Security Services	D-1
Configuring DCE for Use with Open Client and Open Server	D-1
Open Server Applications and DCE Security	D-2
Client-Library Applications and DCE Security	D-3

E. CyberSAFE Kerberos Security Services

Supported Security Services	E-1
Configuring CyberSAFE Kerberos for Use with Open Client and Open Server	E-1
Open Server Applications and CyberSAFE Kerberos	E-2
Client-Library Applications and CyberSAFE Kerberos	E-3

Index

List of Tables

Table 1:	Syntax conventions.....	xvii
Table 2-1:	Open Client Configuration Tasks.....	2-3
Table 3-1:	Open Server Configuration Tasks.....	3-3
Table 4-1:	Server attributes.....	4-2
Table 6-1:	dscp and dscp_dce commands.....	6-1
Table 6-2:	Server attributes.....	6-6
Table A-1:	Environment variables used for connection.....	A-1
Table B-1:	Names and locations for configuration files.....	B-1
Table B-2:	Supported security drivers.....	B-5
Table B-3:	Default transport-type driver mappings.....	B-7
Table C-1:	Environment variables used for localization.....	C-2
Table C-2:	Coded character set conversion modes.....	C-9

About This Book

The *Open Client/Server Configuration Guide for UNIX* contains information about configuring your system to run Open/Client Server™ products on the following platforms:

- HP 9000 HP-UX
- IBM RISC System/6000
- Sun Solaris 2.x (SPARC)

Audience

This book is written for System Administrators. It discusses configuration tasks and topics in terms of system administration rather than application programming.

How to Use This Book

The *Open Client/Server Configuration Guide for UNIX* is divided into the three parts:

- Configuration Instructions
- Configuration Utilities
- Configuration Reference

Configuration Instructions

Part 1, “Configuration Instructions,” contains a chapter for each area of configuration. Each chapter begins with an overview of the configuration process and then gives step-by-step instructions for completing configuration tasks.

- Chapter 1, “Configuration Overview,” provides an overview of the configuration process and configuration requirements.
- Chapter 2, “Basic Configuration for Open Client,” explains how a client application connects to a server and lists the configuration tasks required for connection.
- Chapter 3, “Basic Configuration for Open Server,” explains how an Open Server™ application listens for client connection requests and lists the configuration tasks required for connection.

- Chapter 4, “Using a Directory Service,” which explains how applications get connection information from a directory service and lists configuration tasks required for an application to use a directory service.
- Chapter 5, “Using Security Services,” explains how applications use network-based security services and lists configuration tasks required for an application to use security services.

Configuration Utilities

Part 2, “Configuration Utilities,” describes the utilities that you can use to perform configuration tasks.

- Chapter 6, “Using `dscp` and `dscp_dce`,” explains how to use the `dscp` or `dscp_dce` utilities to configure the server entries in directory services and the interfaces file. `dscp` and `dscp_dce` are command-line utilities.
- Chapter 7, “Using `dsedit` and `dsedit_dce`,” explains how to use the `dsedit` or `dsedit_dce` utility to configure the server entries in directory services and the interfaces file. `dsedit` and `dsedit_dce` are X-Windows utilities with a graphical user interface.

Configuration Reference

Part 3, “Configuration Reference,” serves as a reference for the tasks listed in Part 1, “Configuration Instructions.” Each source of configuration information is described in detail.

The configuration topics are divided into appendices by source of configuration information.

- Appendix A, “Environment Variables,” lists the environment variables that Open Client/Server products use and explains how to set environment variables.
- Appendix B, “Configuration Files,” presents an overview of configuration files and describes:
 - `libtcl.cfg`, the driver configuration file
 - `interfaces`, the interfaces file
 - `ocs.cfg`, the run-time configuration file
- Appendix C, “Localization,” presents an overview of localization files and describes:

- *locales.dat* file
- *objectid.dat* file
- *mnemonic.dat* file
- Conversion configuration files
- Localized message files
- Collating sequence files
- Appendix D, “DCE Security Services,” lists the security services supported by the DCE security driver and summarizes DCE configuration requirements for use as an Open Client/Server security mechanism.
- Appendix E, “CyberSAFE Kerberos Security Services,” lists the security services supported by the CyberSAFE Kerberos security driver and summarizes CyberSAFE configuration requirements for use as an Open Client/Server security mechanism.

Related Documents

- The *Open Client/Server Release Bulletin* contains last-minute information about the release.
- *Installing Sybase Products* contains installation procedures for installing your Open Client/Server software.
- The *Open Client Client-Library Reference Manual* contains reference information for Open Client Client-Library™.
- The *Open Client DB-Library Reference Manual* contains reference information for DB-Library™.
- The *Open Client Client-Library Programmer's Guide* contains information on how to design and implement Client-Library programs.
- The *Open Server Server-Library Reference Manual* contains reference information for Open Server Server-Library.
- The *Open Client and Open Server Common Libraries Reference Manual* contains reference information for CS-Library. CS-Library is a collection of utility routines that are useful in both Client-Library and Server-Library applications.
- The *Open Client/Server Programmer's Supplement* contains platform-specific information for programmers using Open Client/Server products. This document includes information about:

- Compiling and linking an application
- The example programs that are included online with Open Client/Server products
- Routines that have platform-specific behaviors
- The *SQL Server Reference Manual* describes Sybase SQL Server™ commands, datatypes, functions, and system procedures.
- The *Transact-SQL User's Guide* documents Transact-SQL®, Sybase's enhanced version of the relational database language. This manual serves as a textbook for beginning users of the database management system.

Other Sources of Information

Electronic Information Sources

For the most up-to-date information on Sybase products, including information on availability, certifications, bugs, and troubleshooting, refer to Sybase's electronic services:

- AnswerBase™, Sybase's CD-ROM knowledge base
- Sybase OpenLine and PrivateLine, the Sybase forums on CompuServe
- SupportPlusSM Online Services
- SyBooks™-on-the-Web

SupportPlus Online Services

To access SupportPlus Online Services:

1. Connect to the Sybase home page: www.sybase.com.
2. Follow links to Technical Support.
3. Follow links to Sybase Enterprise Technical Support.
4. Follow links to SupportPlus Online Services. If you are not yet registered, first follow links to Register Online for SupportPlus.

SupportPlus Online Services is intended for customer use only. You must, therefore, register to access SupportPlus Online Services.

SyBooks-on-the-Web

To access SyBooks-on-the-Web:

1. Connect to the Sybase home page: www.sybase.com.
2. Follow links to Documentation.

SyBooks-on-the-Web is accessible to the public.

Conventions

This manual uses the following style conventions:

- Commands you should enter exactly as shown appear in monospace font:

`this font`

Words you should replace with the appropriate value for your installation appear in monospace, italic font:

this font

- File names and directories appear in italics:

/usr/u/sybase

- The names of programs, utilities, procedures, and commands appear in narrow, bold font:

dscp

The conventions for syntax statements are shown in Table 1.

Table 1: Syntax conventions

Key	Definition
command	Command names, command option names, utility names, utility flags, and other keywords are in bold .
<i>variable</i>	Variables, or words that stand for values that you fill in, are in <i>italics</i> .
{ }	Curly braces indicate that you choose at least one of the enclosed options. Do not include braces in your option.
[]	Brackets mean choosing one or more of the enclosed options is optional. Do not include brackets in your option.
()	Parentheses are to be typed as part of the command.

Table 1: Syntax conventions (continued)

Key	Definition
	The vertical bar means you can select only one of the options shown.
,	The comma means you can choose as many of the options shown as you like, separating your choices with commas to be typed as part of the command.

If You Need Help

Each Sybase installation that has purchased a support contract has one or more designated people who are authorized to contact Sybase Technical Support. If you cannot resolve a problem using the manuals or online help, ask a designated person at your site to contact Sybase Technical Support.

Please contact the distributor from which your software was purchased if you do not know how to contact Sybase Technical Support.

Part 1: Configuration Instructions

1

Configuration Overview

Welcome to the *Open Client/Server Configuration Guide for UNIX*. Before you read this document, install Open Client™ and/or Open Server according to the instructions in *Installing Sybase Products*.

This chapter gives an overview of the configuration process for Open Client and Open Server. The following topics are included:

- About Open Client and Open Server 1-1
- Overview of Configuration 1-1
- Configuration Tasks 1-3

About Open Client and Open Server

Open Client provides an application programming interface (API) and Net-Library, which enable communications between SQL Server and Open Server applications and customer applications, third-party products, and other Sybase products.

Open Server provides the tools and interfaces needed to create custom servers. Like Open Client, a programming API and Net-Library enable communications with clients and other servers. In addition, Open Server provides routines that:

- Handle multiple client connections
- Schedule interactions with clients
- Handle error conditions
- Perform other functions required from a server

See the following documents for detailed information about Open Client and Open Server:

- *Open Client Client-Library Reference Manual*
- *Open Client DB-Library Reference Manual*
- *Open Server Server-Library Reference Manual*

Overview of Configuration

Open Client/Server software requires specific information in order to function correctly. **Configuration** is the process of setting up your system to make this information available.

Open Client and Open Server use configuration information to:

- Initialize the Open Client or Open Server application
- Establish a connection with SQL Server or an Open Server application

The Initialization Process

To initialize an application, Open Client and Open Server:

Action	Source of Configuration Information
Determine the location of the Sybase installation directory.	<ul style="list-style-type: none"> • SYBASE environment variable
Determine what language, character set, and collating sequence the application uses.	<ul style="list-style-type: none"> • Locale-specific POSIX environment variables LC_*, LANG, LC_ALL, and LC_COLLATE • <i>locales.dat</i> file
Load the directory driver, security driver, and network (Net-Library) driver, as required.	<ul style="list-style-type: none"> • <i>libtcl.cfg</i> file

The Connection Process

Clients and servers communicate via a **connection**. In order for a client application to connect to a server application, the server application must be listening for the client connection request.

To establish a connection, Open Client:

Action	Source of Configuration Information
Determines the name of the target server.	<ul style="list-style-type: none"> • DSQUERY environment variable <p>Uses DSQUERY only if the Open Client application does not specify the name of the server.</p>
Obtains the target server's address.	<ul style="list-style-type: none"> • <i>interfaces</i> file • directory service

To listen for a connection request, Open Server:

Action	Source of Configuration Information
Determines the name of the Open Server application.	<ul style="list-style-type: none"> • DSLISTEN environment variable Uses DSLISTEN only if the Open Server application does not specify a server during initialization.
Obtains the Open Server application's address.	<ul style="list-style-type: none"> • <i>interfaces</i> file • directory service

Configuration Tasks

You must complete some basic configuration tasks in order for an Open Client/Server product to initialize the application and make a connection. These tasks include:

- Setting environment variables to specify a target's default server and initial localization values. The values of DSQUERY and DSLISTEN are used, respectively, if the Open Client and Open Server applications do not explicitly specify a name of a server.
- Ensuring that the target server's address is available
- Configuring your network driver, if needed

There are additional tasks if you are:

- Using a directory service
- Using security services
- Using custom localization values in addition to or instead of initial localization values

The following chapters provide configuration instructions. Refer to the configuration chapter(s) appropriate to your installation.

2

Basic Configuration for Open Client

This chapter discusses the basic configuration requirements for Open Client. The following topics are included:

- Overview of Basic Configuration 2-1
- Configuration Tasks 2-3

► **Note**

Except where noted, information in this chapter applies to both DB-Library and Client-Library.

Specifically, DB-Library does not use environment variables to determine initial localization values and does not examine the *libtcl.cfg* file. However, DB-Library does examine the SYBASE and DSQUERY environment variables.

For more information on DB-Library, see the *Open Client DB-Library/C Reference Manual*.

Overview of Basic Configuration

All Open Client applications require some basic configuration information, obtained during initialization and connection. These basic configuration items are:

► **Note**

Items 2, 6, and 7 do not apply to DB-Library.

► **Note**

Items 1 through 3 occur when the Open Client Client-Library application calls the `cs_ctx_alloc` or `cs_ctx_global` routine. Items 4 through 6 occur when the Open Client application calls `ct_connect`.

1. The location of the Sybase installation directory as defined by the SYBASE environment variable.
2. A locale name. Open Client uses the values of the following POSIX environment variables as locale names:
 - LC_ALL
 - LANG, if LC_ALL is not definedOpen Client later uses this value to obtain localization information from the *locales.dat* file. If neither environment variable is defined, Open Client uses “default” as the locale name.
3. Localized message and character set files. Open Client looks in the *locales.dat* file for an entry whose name matches the locale name determined in step 2. Then it loads the localized messages and character set files specified in the *locales.dat* file.
4. The name of the target server. Open Client obtains the name of the target server from one of the following sources, in the order listed:
 - The client application, which can provide the server name in the call to `ct_connect` (or `dbopen`). Some applications, such as `isql`, can specify the name of the target server through command line options.
 - The DSQUERY environment variable, if the application does not specify the target server.
 - The default name SYBASE, if DSQUERY is not set.
5. The target server’s network address. Open Client gets the target server’s addresses from the directory service or from *interfaces*.

Directory service – Open Client looks for an entry in the [DIRECTORY] section of *libtcl.cfg* to determine where to look up server address information. The setting of the CS_DS_PROVIDER property determines which [DIRECTORY] entry the application searches for, or defaults to the first entry of the [DIRECTORY] section.

interfaces – If a directory service is not used, or if it is used and fails, Open Client searches for the SERVERNAME entry in *interfaces* that matches the name as determined in step 4 and uses the corresponding target address.
6. The name of the network driver. Open Client looks in the [DRIVERS] section of *libtcl.cfg* to determine which network driver to load.

Table 6-2 on page 6-6 under “Transport Type” lists valid network protocols. This value determines which network driver to load. “Adding a Network Driver” on page B-9 gives instructions on adding a network driver.

7. The name of the security service driver. Open Client looks in the [SECURITY] section of *libtcl.cfg* to determine which security driver to load.

Refer to Chapter 5, “Using Security Services” for more information on security services.

Configuration Tasks

To enable Open Client to successfully initialize your client application and carry out connection requests, complete these tasks:

Table 2-1: Open Client Configuration Tasks

Task	For More Information
<p>Set environment variables:</p> <ul style="list-style-type: none"> • Set the LC_ALL or LANG environment variable to the desired locale name. <p>The locale name you specify must correspond to an entry in <i>locales.dat</i>. If you do not set LC_ALL or LANG, make sure that the “default” entry in <i>locales.dat</i> reflects the localization values your applications will use.</p> <p>Ensure that you have localization files that match the language, character set, and collating sequence specified in the locales file.</p> <ul style="list-style-type: none"> • If your application uses custom localization values, set the LC_ALL, LC_COLLATE, LC_TYPE, LC_MESSAGE, or LC_TIME environment variable to the locale name. <p>If you do not know which environment variable your application uses, set all the environment variables to the desired locale name.</p> <ul style="list-style-type: none"> • Set the DSQUERY environment variable to the name of the target server. <p>If the client application names the target server, you do not need to set DSQUERY. If DSQUERY is not set and the application does not name the server, Open Client uses the server name “SYBASE.”</p>	<p>See Appendix A, “Environment Variables,” for instructions about how to set environment variables.</p> <p>See Appendix C, “Localization,” for information on localization.</p>

Table 2-1: Open Client Configuration Tasks (continued)

Task	For More Information
<p>Configure <i>libtcl.cfg</i>:</p> <ul style="list-style-type: none"> • Specify a network transport driver in the [DRIVERS] section of <i>libtcl.cfg</i>. • Specify a directory driver in the [DIRECTORY] section of <i>libtcl.cfg</i>. • Specify a security driver in the [SECURITY] section of <i>libtcl.cfg</i>. 	<p>See Appendix B, “Configuration Files” for reference information about <i>libtcl.cfg</i>.</p>
<p>Configure <i>interfaces</i> or directory service:</p> <ul style="list-style-type: none"> • Create a server entry in <i>interfaces</i> or DCE directory service using <i>dscp</i> or <i>dscp_dce</i> respectively. 	<p>See Chapter 6, “Using <i>dscp</i> and <i>dscp_dce</i>,” for instructions about using <i>dscp</i> and <i>dscp_dce</i>.</p> <p>See “<i>interfaces</i>” on page B-10 for information about <i>interfaces</i>.</p> <p>See “Using a Directory Service” on page 4-1 for information about directory services.</p>

3

Basic Configuration for Open Server

This chapter discusses the basic configuration requirements for Open Server. The following topics are included:

- About Open Server Applications 3-1
- Overview of Basic Configuration 3-1
- Configuration Tasks 3-3

About Open Server Applications

Open Server applications fall into three functional categories:

- Standalone
- Auxiliary
- Gateway

The configuration of an Open Server application depends on which category it falls into. See the *Open Server Server-Library/C Reference Manual* for more information about the types of Open Server applications.

Overview of Basic Configuration

All Open Server applications require some basic configuration information, obtained during initialization and connection. These basic configuration items are:

1. The location of the Sybase installation directory as defined by the SYBASE environment variable.
2. A locale name. Open Server uses the values of the following POSIX environment variables as locale names:
 - LC_ALL
 - LANG, if LC_ALL is not defined

Open Server later uses this value to obtain localization information from *locales.dat*. If neither environment variable is defined, Open Server uses “default” as the locale name.

3. Localized message and character set files. Open Server looks in *locales.dat* for an entry whose name matches the locale name

determined in step 2. Then it loads the localized messages and character set files specified in *locales.dat*.

4. The name of the target server. Open Server obtains the name of the Open Server application from one of the following sources, in the order listed:
 - The Open Server application, which can provide the server name in the call to *srv_init*.
 - The *DSLISTEN* environment variable, if the application does not specify its name.
 - The default name *SYBASE*, if *DSLISTEN* is not set.
5. The target server's network address. Open Server gets the target server's address(es) from either the directory service or from *interfaces*.

Directory service – Open Server looks for an entry in the [DIRECTORY] section of *libtcl.cfg* to determine where to look up server address information. The setting of the *CS_DS_PROVIDER* property determines which [DIRECTORY] entry the application searches for, or defaults to the first entry of the [DIRECTORY] section.

interfaces – If a directory service is not used, or if it is used and fails, Open Server searches for the *SERVERNAME* entry in *interfaces* that matches the name as determined in item 4 and uses the corresponding target address.
6. When a client requests a connection that uses a network-based security mechanism, Open Server looks up the corresponding security driver in the [SECURITY] section of *libtcl.cfg*.

Configuration Tasks

To enable Open Server to successfully initialize your server application and respond to connection requests, complete these tasks:

Table 3-1: Open Server Configuration Tasks

Task	For More Information
<p>Configure <i>libtcl.cfg</i>:</p> <ul style="list-style-type: none"> Specify a directory driver in the [DIRECTORY] section of <i>libtcl.cfg</i>. Specify a security driver in the [SECURITY] section of <i>libtcl.cfg</i>. 	<p>See Appendix B, "Configuration Files" for reference information about <i>libtcl.cfg</i>.</p>
<p>Configure <i>interfaces</i> or directory service:</p> <ul style="list-style-type: none"> Create a server entry in <i>interfaces</i> or DCE directory service using <i>dscp</i> or <i>dscp_dce</i> respectively. 	<p>See Chapter 6, "Using <i>dscp</i> and <i>dscp_dce</i>," for instructions about using <i>dscp</i> and <i>dscp_dce</i>.</p> <p>See "interfaces" on page B-10 for reference information about <i>interfaces</i>.</p> <p>See "Using a Directory Service" on page 4-1 for information about directory services.</p>

Table 3-1: Open Server Configuration Tasks (continued)

Task	For More Information
<p>Set environment variables:</p> <ul style="list-style-type: none"> • Set the LC_ALL or LANG environment variable to the desired locale name. <p>The locale name you specify must correspond to an entry in <i>locales.dat</i>. If you do not set LC_ALL or LANG, make sure that the “default” entry in <i>locales.dat</i> reflects the localization values your applications will use.</p> <p>Ensure that you have localization files that match the language, character set, and collating sequence specified in <i>locales</i>.</p> <ul style="list-style-type: none"> • If your application uses custom localization values, set the LC_ALL, LC_COLLATE, LC_TYPE, LC_MESSAGE, or LC_TIME environment variable to the locale name. <p>If you do not know which environment variable your application uses, set all the environment variables to the desired locale name.</p> <ul style="list-style-type: none"> • Set the DSLISTEN environment variable to the name of the Open Server application. <p>If the name of the Open Server application is coded into the application, you do not need to set DSLISTEN. If DSLISTEN is not set and the application does not name the server, Open Server uses the server name “SYBASE.”</p> <ul style="list-style-type: none"> • If the Open Server application acts as a gateway application, set the DSQUERY environment variable to the name of the target server. 	<p>See Appendix A, “Environment Variables,” for instructions about how to set environment variables.</p> <p>See Appendix C, “Localization,” for information on localization.</p>

4

Using a Directory Service

Client-Library and Server-Library applications can use directory services to keep track of information about servers. This chapter discusses how a directory service works and what you need to configure to use one. The following topics are included:

- Overview of Directory Services 4-1
- How Applications Use a Directory Service 4-3
- Configuration Tasks 4-5

► *Note*

DB-Library does not support directory services.

Overview of Directory Services

A **directory service** manages the creation, modification, and retrieval of information about network entities. Client-Library and Server-Library applications can use a directory service as an alternative to *interfaces* to obtain information about servers.

The advantage of using a directory service is that you do not need to update multiple *interfaces* files when a new server is added to your network or when a server moves to a new address.

UNIX platforms can use the Cell Directory Service (CDS) provided by Distributed Computing Environment (DCE).

Server Objects and Attributes

The directory service must contain information about servers accessed by your Open Client. You can add or modify the information in a directory service using `dscp_dce`. Use `dscp` to modify *interfaces*.

A directory service identifies a server entry as a **directory object**. Each directory object has a unique set of **attributes**. Client-Library and Server-Library recognize these attributes:

Table 4-1: Server attributes

Attribute	Description
Server Object Version	Symbolic integer code for the version of the object definition. Sybase provides this attribute to identify future changes to the object definition.
Server Name	A string value that specifies the server's name. The name can be any string that is 512 or fewer bytes long. Do not confuse a server name attribute with the name used to locate the directory entry. The latter is the fully qualified name for the directory entry, expressed in the name syntax of the directory provider. To avoid confusion, administrators should ensure that the name attribute at least partially matches the server's fully-qualified name (for example, the attribute value could be the entry's common name).
Server Service	A string value that describes the service that the server provides. The service value can be any string that is 512 or fewer bytes long.
Server Status	Symbolic integer code that describes the operating status of the server. Valid values are: <ul style="list-style-type: none"> • 1: Active • 2: Stopped • 3: Failed • 4: Unknown
Transport Address	One or more transport addresses for the server. The transport address attribute has two elements: <ul style="list-style-type: none"> • Transport type • Transport address
Security Mechanism	Object identifier strings (OID) that specify the security mechanisms supported by the server. This attribute is optional. If it is omitted, the Open Server allows clients to connect with any security mechanism for which the Open Server has a corresponding security driver. (See "Server Library and Security Services" on page 5-5 for process details.) See "objectid.dat" on page C-12 for more information about object identifier strings. See the [secmech] section of <i>SSYBASE/config/objectid.dat</i> for examples.

How Applications Use a Directory Service

Client-Library and Server-Library can use a directory service, rather than *interfaces*, to obtain a server's address.

Open Client/Server software uses a directory driver to retrieve information from a directory service.

A directory driver is a Sybase library that provides Open Client/Server software with a generic interface to a specific directory service. Sybase provides a directory driver for each supported directory service.

Client-Library and Server-Library determine whether to use a directory service or *interfaces* as follows:

1. If the application specifies a directory driver, Client-Library by calling `ct_con_props` (`CS_SET`, `CS_DS_PROVIDER`) and Server-Library by calling `srv_props` (`CS_SET`, `SRV_S_DSPROVIDER`), the application checks in the `DIRECTORY` section of `libtcl.cfg` for a matching driver and loads that driver.
See "libtcl.cfg" on page B-1 for reference information about directory drivers and `libtcl.cfg`.
2. If the client application does not specify a directory driver, Client-Library and Server-Library load the directory driver listed by the first entry in the `[DIRECTORY]` section of `libtcl.cfg`.
3. Client-Library and Server-Library fall back and use *interfaces* to obtain the server's address if any of the following are true:
 - `libtcl.cfg` does not exist.
 - There are no entries in the `[DIRECTORY]` section of `libtcl.cfg`.
 - The specified directory driver fails to load.

In Figure 4-1, the Open Client application uses a directory service to locate and connect to the Open Server *my_server*:

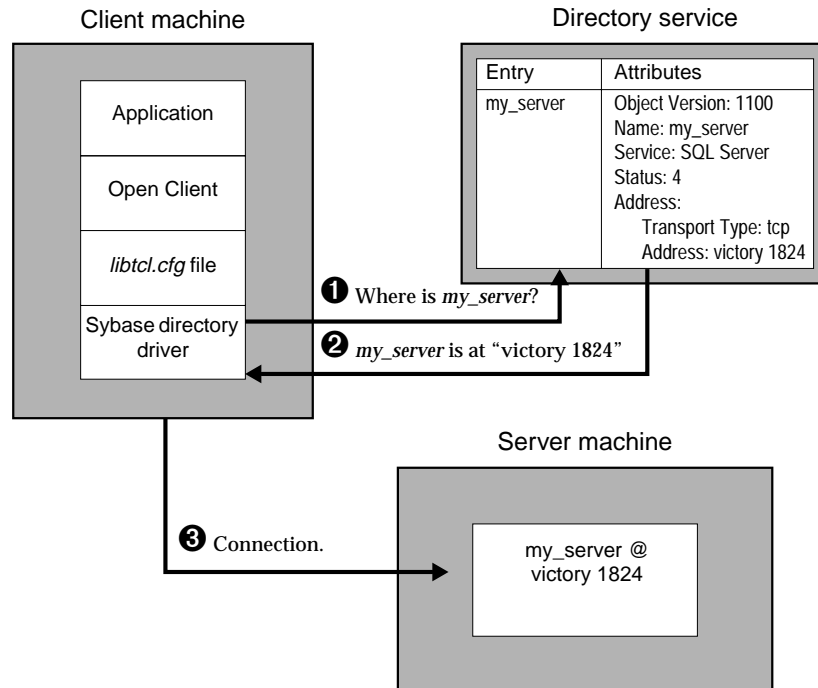


Figure 4-1: Client/server connection using a directory service

The connection process shown in Figure 4-1 in three steps:

1. Client-Library uses the Sybase directory driver specified in the *libtcl.cfg* file to request the address of *my_server*.
2. The directory service looks up the attributes for the *my_server* entry and returns the information to Client-Library via the Sybase directory driver.
3. The application uses the address to connect to the machine where *my_server* resides.

Configuration Tasks

To enable Client-Library and Server-Library applications to use a directory service, complete these tasks:

Task	For More Information
Configure <i>libtcl.cfg</i>: <ul style="list-style-type: none">• Specify a directory driver and DIT base in the DIRECTORY section of <i>libtcl.cfg</i>.• Make sure that the DIT base exists in the DCE directory structure.• Open Client/Server software use the first entry in the DIRECTORY section as the default directory driver.	See “libtcl.cfg” on page B-1 for information about directory drivers and <i>libtcl.cfg</i> .
Configure the directory service: <ul style="list-style-type: none">• Create an entry for the target server in the directory service using the appropriate utility.	See Chapter 6, “Using dscp and dscp_dce,” for instructions on using dscp and dscp_dce. See Chapter 7, “Using dsedit and dsedit_dce,” for instructions on using dsedit and dsedit_dce.

5

Using Security Services

Client-Library and Server-Library applications can use the security services provided by third-party security software to authenticate users and protect data transmitted between machines on a network.

► *Note*

Network-based security is new for Client-Library release 11.1 and requires a server based on Open Server release 11.1 or later. Open Client DB-Library and SQL Server 11.0 do not support network-based security services.

This chapter discusses how network-based security works and what you need to configure to use it. The following topics are included:

- Overview of Network-Based Security 5-1
- How Applications Use Security Services 5-3
- Configuration Tasks 5-6

Overview of Network-Based Security

In a distributed client/server computing environment, intruders can view or tamper with confidential data. Network-based security takes advantage of third-party distributed security software to authenticate users and protect data transmitted between machines on a network.

Security Mechanisms

Sybase defines a **security mechanism** as external software that provides security services for a connection. UNIX platforms can use the security mechanism provided by Distributed Computing Environment (DCE), as well as kerberos security services provided by CyberSAFE Challenger.

You can specify the security mechanisms that a server supports in *interfaces* or a directory service. The values for *interfaces* or directory service's *secmech* line/attribute must correspond to the strings associated with object identifiers defined in the user's *objectid.dat* file, under the [secmech] section:

- The `secmech` line in an *interfaces* entry specifies the security mechanisms that a server supports. The `secmech` line is optional.
- The `secmech` attribute in a directory service entry describes the security mechanisms that a server supports. This attribute is optional.

When a client gets the server's address, it can verify that the server supports the security mechanism that the client is using:

- If there is a `secmech` line or attribute and security mechanisms are listed, then only those security mechanisms are allowed.
- If there is no `secmech` line or attribute, then all security mechanisms are allowed.
- If there is a `secmech` line or attribute but no security mechanisms are listed, then the server does not support any security mechanisms.

Security Drivers

Sybase provides **security drivers** that allow Client-Library and Server-Library to communicate with the security mechanism. Each Sybase security driver maps a generic interface to the security provider's interface.

In order to use a security mechanism on a connection, both items below must be true:

- The client and server must use compatible security drivers. For example, a client using a DCE driver requires a server using a DCE driver.
- The client application must request services by setting connection properties before connecting to the server.

Security Services

Each security mechanism provides a set of **security services** used to establish a secure connection between a client and a server. Each security service addresses a particular security concern.

Security services can be divided into two broad categories:

- Authentication services
- Per-packet security services

See the *Open Client Client-Library/C Reference Manual* for a complete discussion of security services.

Client-Library applications set connection properties to request a mechanism's services. Open Server applications read the properties of a client thread to determine which services are being performed.

See Appendix D, "DCE Security Services," and Appendix E, "CyberSAFE Kerberos Security Services," for a list of security services provided by DCE and CyberSAFE Challenger.

How Applications Use Security Services

Client-Library and Server-Library applications can use a security mechanism to perform authentication and per-packet security services. The security mechanism behaves like a clearinghouse through which Client-Library and Server-Library validate information.

Figure 5-1 shows a client application using a security mechanism to ensure a secure connection with a server.

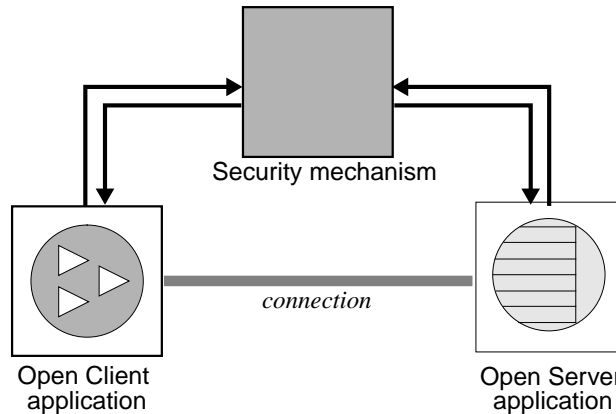


Figure 5-1: Open Client and Open Server applications using a security mechanism

Figure 5-1 applies to both authentication and per-packet security services.

If an Open Client application requests authentication services:

1. Client-Library validates the login with the security mechanism. The security mechanism returns a login record, or **token**.
The security mechanism creates the login token based on which security services are requested.
2. Client-Library establishes a transport connection with the Open Server application and sends its login token.
3. Server Library authenticates the client's login token with the security mechanism. If the login is valid, the Open Server application establishes a secure connection.

If an Open Client application requests per-packet security services:

1. Client-Library uses the security mechanism to prepare the data packet it will send to the Open Server application.
Depending on which security services are requested, the security mechanism might encrypt the data or create a cryptographic signature associated with the data.
2. Client-Library sends the data packet to the Open Server application.
3. When Open Server receives the data packet, it uses the security mechanism to perform any required decryption and validation.
Refer to the "Security Features" Topics page in the *Open Client Client-Library/C Reference Manual* for a detailed explanation of Client-Library's security features.

Client-Library and Security Services

You can set connection properties in Open Client applications to request a security mechanism and the security mechanism's services. Client-Library determines which security mechanism and services to use on the connection as follows:

1. If the client application specifies a security mechanism, Client-Library checks in the SECURITY section of *libtcl.cfg* for a matching driver and loads that driver.
2. If the client application does not specify a security driver, Client-Library loads the security driver listed by the first entry in the [SECURITY] section of *libtcl.cfg*.
3. Client-Library determines which security services will be used for the connection from the client application.

If there is no *libtcl.cfg* or there are no entries in the [SECURITY] section, then there is no network security provider. In that case, the Open Server application authenticates the user if the user supplies the correct password.

Server Library and Security Services

Open Server applications can read the properties of a client connection request to determine which security mechanism to use and which services to perform.

By default, an Open Server application supports the security mechanisms listed in the [SECURITY] section of *libtcl.cfg*. Administrators can further restrict the list of supported mechanisms by adding a “secmech” attribute to the server’s directory entry or a “secmech” line to the Open Server application’s *interfaces* entry.

When an Open Client application requests a security session from an Open Server application:

1. Server Library reads the security token that was sent with the client connection request.

The security token contains the object identifier for the security mechanism that the client uses.

2. If the Open Server application’s *interfaces* entry or directory service entry lists the secmech line/attribute, Server Library searches the secmech line/attribute for a value corresponding to the object identifier specified in the security token.

If a matching value is not found, the connection request is rejected.

3. Server Library searches *objectid.dat* to match the object identifier with the local name of the security mechanism.

See Appendix C, “Localization,” for reference information about *objectid.dat*.

4. Server Library loads the security driver associated with the local name of the security mechanism.

The security driver is listed in the [SECURITY] section of *libtcl.cfg*.

Configuration Tasks

To enable your Open Client/Server application to use security services, complete these tasks:

Task	For More Information
<p>Configure <i>libtcl.cfg</i>:</p> <p>Specify a security driver in the [SECURITY] section of <i>libtcl.cfg</i>.</p> <p>Note: Open Client/Server software use the first entry in the [SECURITY] section as the default security driver.</p>	<p>See “<i>libtcl.cfg</i>” on page B-1 for reference information about security drivers and <i>libtcl.cfg</i>.</p>
<p>Configure DCE or CyberSAFE Kerberos to work with Open Client and Open Server</p>	<p>For DCE information, see Appendix D, “DCE Security Services,” and your DCE documentation.</p> <p>For CyberSAFE information, see Appendix E, “CyberSAFE Kerberos Security Services,” and your CyberSAFE documentation.</p>

Optionally, to restrict the security mechanisms that a server supports, complete this task:

Task	For More Information
<p>Restrict security mechanisms that the server supports:</p> <ul style="list-style-type: none"> • If your application uses <i>interfaces</i>, add a <i>secmech</i> line in the server’s <i>interfaces</i> entry using <i>dscp</i>. • If your application uses a directory service, add the <i>secmech</i> attribute to the server’s directory service entry using <i>dscp_dce</i>. 	<p>See Chapter 6, “Using <i>dscp</i> and <i>dscp_dce</i>,” for information about adding information to a directory service or <i>interfaces</i>.</p>

Part 2: Configuration Utilities

6

Using *dscp* and *dscp_dce*

This chapter explains how to use *dscp* to configure the *interfaces* file and *dscp_dce* to configure a directory service. The following topics are included:

- About *dscp* and *dscp_dce* 6-1
- Starting *dscp* and *dscp_dce* 6-3
- Getting Help 6-3
- Using *dscp* and *dscp_dce* Sessions 6-4
- Adding and Modifying Server Entries 6-5
- Copying Server Entries 6-10
- Exiting *dscp* and *dscp_dce* 6-12

About *dscp* and *dscp_dce*

dscp and *dscp_dce* are command line utilities that allow you to view and edit server entries in *interfaces* or a DCE directory service, respectively. After opening a session, you can check your configuration, view existing entries, create new entries, and modify entries as needed.

Both utilities allow you to perform functions by entering commands at the *dscp* or *dscp_dce* prompt. Unless otherwise noted, *dscp* and *dscp_dce* commands perform the same functions. Table 6-1 describes these commands:

Table 6-1: *dscp* and *dscp_dce* commands

Command	Description
<code>open [DSNAME]</code>	Opens a session with the specified directory service or <i>interfaces</i> . <i>dscp</i> – To open a session with <i>interfaces</i> , specify “InterfacesDriver” as <i>DSNAME</i> . <i>dscp_dce</i> – <i>DSNAME</i> must match a directory service provider as listed in <i>libtcl.cfg</i> . If you do not specify <i>DSNAME</i> , <i>dscp_dce</i> defaults to the first directory service provider listed in <i>libtcl.cfg</i> .
<code>sess</code>	Lists all open sessions.

Table 6-1: dscp and dscp_dce commands (continued)

Command	Description
[switch] <i>SESS</i>	Makes session number <i>SESS</i> the current session.
close [<i>SESS</i>]	Closes a session identified by the <i>SESS</i> number. If you do not specify <i>SESS</i> , closes the current session.
list [all]	Lists the server entries for the current session. To list the names of the entries, use the list command. To list the attributes for each entry, use the list all command.
read <i>SERVERNAME</i>	Prints the contents of server entry <i>SERVERNAME</i> to the screen.
add <i>SERVERNAME</i>	Adds server entry <i>SERVERNAME</i> in the current session. <i>dscp</i> and <i>dscp_dce</i> prompt you for information about <i>SERVERNAME</i> . Press Return to accept the default value, which is shown in brackets [].
addattr <i>SERVERNAME</i>	Adds an attribute to the server entry <i>SERVERNAME</i> in the current session.
mod <i>SERVERNAME</i>	Modifies server entry <i>SERVERNAME</i> in the current session. <i>dscp</i> and <i>dscp_dce</i> prompt you for information about <i>SERVERNAME</i> . Press Return to accept the default value, which is shown in brackets [].
del <i>SERVERNAME</i>	Deletes server entry <i>SERVERNAME</i> in the current session.
delete-all	Deletes all server entries in the current session.
copy <i>NAME1</i> to { <i>NAME2</i> <i>SESS</i> <i>SESS NAME2</i> }	Copies server entry <i>NAME1</i> in the current session to: <ul style="list-style-type: none"> • Server entry <i>NAME2</i> in the current session • Session <i>SESS</i> • Server entry <i>NAME2</i> in session <i>SESS</i>
copyall to <i>SESS</i>	Copies all server entries in the current session to session <i>SESS</i> .
config	Prints configuration information related to your Sybase environment to the screen.
exit, quit	Exits <i>dscp</i> or <i>dscp_dce</i> .
help, ?, h	Displays the help screen.

Starting *dscp* and *dscp_dce*

If you plan to add or modify entries, you must log into the directory service, with the necessary privileges, before you start *dscp* or *dscp_dce*.

To start *dscp*, enter:

```
$SYBASE/bin/dscp
```

The *dscp* prompt, `>>`, appears.

To start *dscp_dce*, enter:

```
$SYBASE/bin/dscp_dce
```

The *dscp_dce* prompt, which is the same as the *dscp* prompt, `>>`, appears.

Viewing Your Configuration

You can view the current Open Client/Server configuration and directory service provider names using the `config` command.

When you enter:

```
config
```

at the prompt, *dscp* and *dscp_dce* prints the following information to the screen:

- The value of the SYBASE environment variable
- The location of the driver configuration file
- The names of directory service providers with which you can open a *dscp* or *dscp_dce* session

Getting Help

To view the *dscp* or *dscp_dce* help screen, enter one of these commands at the prompt:

```
help
```

```
h
```

```
?
```

Using *dscp* and *dscp_dce* Sessions

Before you can view, add, or modify server entries, you must open a session.

Opening a *dscp_dce* session allows you to interact with any directory service which has a driver listed in *libtcl.cfg*.

Opening a *dscp* session allows you to interact with *interfaces*.

You can have multiple sessions open at one time.

Opening a Session

To open a session, enter:

```
open DSNAME
```

at the prompt. For example, if you enter:

```
open dce1
```

dscp_dce opens a session with the directory service associated with the “dce1” directory driver. If you do not specify *DSNAME*, *dscp_dce* uses the default directory service provider.

To open a session with *interfaces*, enter:

```
open InterfacesDriver
```

at the prompt.

When you open a session, *dscp* and *dscp_dce* tells you the session’s number. For example, if you open a session with *interfaces* using the `open InterfacesDriver` command, *dscp* returns the following message:

```
ok  
Session 1 InterfacesDriver>>
```

Listing Sessions

To list all open sessions, enter:

```
sess
```

Switching Between Open Sessions

To switch to another open session, enter:

```
switch SESS
```

at the prompt.

where:

SESS is the session number.

For example, if you enter:

```
switch 3
```

you are switched to session 3. The `switch` keyword is optional. For example, entering:

```
3
```

also switches you to session 3.

Closing a Session

To close a session, enter

```
close SESS
```

at the prompt.

where:

SESS is the session number.

For example, if you enter:

```
close 3
```

session 3 is closed. Use the `sess` command to list all open sessions.

If you do not specify *SESS*, the current session is closed.

Adding and Modifying Server Entries

Once you open a session with a directory service or *interfaces*, you can list, add, modify, and delete server entries associated with that session.

► **Note**

When you add or modify a server entry, `dscp` automatically creates or modifies both master and query lines. The master line and the query line of an *interfaces* file entry contain identical information.

Each server entry is made up of a set of attributes. When you add or modify a server entry, `dscp` and `dscp_dce` prompt you for information about each attribute. Table 6-2 describes each attribute:

Table 6-2: Server attributes

Attribute	Type of Value	Default Value	Capable of being Modified when adding or modifying server entry										
Server Object Version	Integer	110	Adding Directory Services: No <i>interfaces: No</i> Modifying Directory Services: Yes <i>interfaces: No</i>										
Server Name	Character string	N/A	Adding Directory Services: N/A <i>interfaces: N/A</i> Modifying Directory Services: No <i>interfaces: No</i>										
Server Service	Character string	SQL SERVER	Adding Directory Services: Yes <i>interfaces: Yes</i> Modifying Directory Services: Yes <i>interfaces: No</i>										
Server Status	Integer	4 Valid values are: <table data-bbox="779 1302 1023 1428"> <thead> <tr> <th>Value</th> <th>Server status</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Active</td> </tr> <tr> <td>2</td> <td>Stopped</td> </tr> <tr> <td>3</td> <td>Failed</td> </tr> <tr> <td>4</td> <td>Unknown</td> </tr> </tbody> </table>	Value	Server status	1	Active	2	Stopped	3	Failed	4	Unknown	Adding Directory Services: No <i>interfaces: No</i> Modifying Directory Services: Yes <i>interfaces: No</i>
Value	Server status												
1	Active												
2	Stopped												
3	Failed												
4	Unknown												

Table 6-2: Server attributes (continued)

Attribute	Type of Value	Default Value	Capable of being Modified when adding or modifying server entry
Transport Address • Transport type • Transport address	Transport type: Character string Transport address: Character string	Transport type: tcp Transport address: None Valid values are: Transport type: "tcp", "spx", "decnet", "tli tcp", "tli spx" Transport address: Character string in a format recognized by the specified transport type	Adding or modifying Directory Services: Transport type: Yes Transport address: Yes <i>interfaces:</i> Transport type: Yes Transport address: Yes
Security Mechanism	Character string Note: You can add up to 20 security mechanism strings for each server entry.	None Valid values are Character strings associated with object identifiers defined in the user's <i>objectid.dat</i> .	Adding Directory Services: Yes <i>interfaces:</i> Yes Modifying Directory Services: Yes <i>interfaces:</i> Yes

Listing Server Entries

To list the names of server entries associated with a session, enter:

```
list
```

at the prompt.

To list the attributes of server entries associated with a session, enter:

```
list all
```

at the prompt.

See Table 6-2 for a description of server attributes.

Viewing a Server Entry

To view the contents of a server entry, enter:

```
read SERVERNAME
```

at the prompt. For example, if you enter:

```
read myserver
```

the following information is displayed on the screen:

```
DIT base for object: interfaces
Distinguish name: myserver
Server Version: 1
Server Name: myserver
Server Service: SQL Server
Server Status: 4 (Unknown)
Server Address:
  Transport Type: tcp
  Transport Addr: victory 1824
  Transport Type: tcp
  Transport Addr: victory 1828
```

See Table 6-2 for a description of the server attributes listed above.

Adding a Server Entry

To add a server entry, enter:

```
add SERVERNAME
```

at the prompt. `dscp` and `dscp_dce` prompt you for information about *SERVERNAME*. Enter a value for each attribute or press return to accept the default value, which is shown in brackets [].

For example, if you enter:

```
add myserver
```

`dscp` or `dscp_dce` prompts you for information as follows:

```
Service: [SQL Server]
Transport Type: [tcp] tcp
Transport Address: victory 8001
Security Mechanism []:
```

To exit add mode, enter:

```
#done
```

at the prompt.

A server entry can have up to 20 transport type/address combinations associated with it.

See Table 6-2 for a description of the server attributes listed above.

Modifying a Server Entry

To modify an existing server entry, enter:

```
mod SERVERNAME
```

at the prompt. `dscp` and `dscp_dce` prompt you for information about *SERVERNAME*. Enter a value for each attribute or press return to accept the existing value, which is shown in brackets [].

For example, if you enter:

```
mod myserver
```

`dscp` or `dscp_dce` prompts you for information as follows:

```
Version: [1]
Service: [SQL Server] Open Server
Status: [4]
Address:
  Transport Type: [tcp]
  Transport Address: [victory 1824] victory 1826
  Transport Type: [tcp]
  Transport Address: [victory 1828]
  Transport Type: []
  Security Mechanism []:
```

► **Note**

`dscp` cannot modify the Version, Service, and Status entries.

To delete an address, enter:

```
#del
```

at the prompt. To exit modify mode, enter:

```
#done
```

at the prompt.

See Table 6-2 for a description of the server attributes listed above.

Deleting Entries

You can delete one entry or all entries associated with a session. To delete one entry, enter:

```
del SERVERNAME
```

at the prompt. For example, if you enter:

```
del myserver
```

`dscp` or `dscp_dce` deletes the entry for “myserver.” To delete all entries associated with a session, enter:

`delete-all`
at the prompt.

Copying Server Entries

`dscp` and `dscp_dce` allow you to copy server entries within a session and between sessions. This includes copying entries from *interfaces* to a directory service.

There are four options when copying a server entry. You can:

- Copy a server entry to a new name in the current session
- Copy a server entry to a different session
- Copy a server entry to a new name in a different session
- Copy all entries in the current session to a different session

Copying Entries Within a Session

You can copy a server entry within a session, if you want to create a new server entry. To copy an entry within a session, enter:

```
copy NAME1 to NAME2
```

at the prompt. For example, if you enter:

```
copy myserver to my_server
```

`dscp` creates a new entry “my_server” identical to “myserver.” You can then modify the new entry and leave the original intact.

Copying Entries Between Sessions

There are two options for copying a server entry between sessions. You can:

- Keep the name of the server entry
- Rename the server entry

To copy an entry to a different session and keep the servername, enter:

```
copy NAME1 to SESS
```

at the prompt.

where:

NAME1 is the current server name

SESS is the number of the session to which you want to copy the server entry.

For example, if you enter:

```
copy myserver to 2
```

dscp or dscp_dce copies the “myserver” entry in the current session to session 2.

To copy an entry to a different session and give it a different name, enter:

```
copy NAME1 to SESS NAME2
```

at the prompt.

where:

NAME1 is the current server name.

SESS is the number of the session to which you want to copy the server entry.

NAME2 is the new server name.

For example, if you enter:

```
copy myserver to 2 my_server
```

dscp or dscp_dce copies the “myserver” entry in the current session to session 2 and renames it “my_server.”

Copying All Entries to a Different Session

To copy all entries in the current session to a different session, enter:

```
copyall SESS
```

at the prompt.

where:

SESS is the number of the session to which you want to copy all entries.

For example, if you enter:

```
copyall 2
```

dscp or dscp_dce copies all entries in the current session to session 2.

Exiting *dscp* and *dscp_dce*

To exit *dscp* or *dscp_dce*, enter one of these commands:

exit

quit

at the prompt.

7

Using *dsedit* and *dsedit_dce*

This chapter explains how to use *dsedit* to configure *interfaces* and *dsedit_dce* to configure Sybase server listings in a directory service. The following topics are included:

- About *dsedit* and *dsedit_dce* 7-1
- Starting *dsedit_dce* and *dsedit_dce* 7-1
- Getting Help 7-2
- Opening a Session 7-2
- Adding, Viewing, and Editing Server Entries 7-6
- Troubleshooting *dsedit* or *dsedit_dce* Problems 7-8

About *dsedit* and *dsedit_dce*

dsedit and *dsedit_dce* are X-Windows based graphical tools that lets you view and edit server entries in *interfaces* or a DCE directory service. *dsedit* edits only *interfaces* entries. *dsedit_dce* edits *interfaces* entries and DCE directory entries.

If your system does not have X-Windows, use *dscp* or *dscp_dce* to configure server entries in *interfaces* or DCE directory. See Chapter 6, “Using *dscp* and *dscp_dce*,” for more information.

Starting *dsedit_dce* and *dsedit_dce*

If you plan to add or modify servers, make sure that you will be able to edit *interfaces* or DCE directory before starting *dsedit* and *dsedit_dce*:

- To edit *interfaces* entries, you must have write permission on *interfaces*.
- To edit DCE directory entries, you must be logged into DCE as a user with administrator privileges.

To start *dsedit*, enter:

```
$SYBASE/bin/dsedit
```

To start *dsedit_dce*, enter:

```
$SYBASE/bin/dsedit_dce
```

If you are running `dsedit` or `dsedit_dce` from a remote machine, make sure that the `DISPLAY` environment variable is set correctly. See your X11 documentation for information on setting the `DISPLAY` environment variable.

Getting Help

On any screen, click the Help button to get help.

Opening a Session

After starting `dsedit` or `dsedit_dce`, you will see the main screen. This screen lets you select and open editing sessions for *interfaces* files or DCE directories.

Interfaces file Sessions

To open the default *interfaces* for editing, select Sybase *interfaces* file and click OK. To open an alternate file, edit the displayed file name before clicking OK. You can open multiple *interfaces* file sessions with different files. Figure 7-1 shows the screen for an *interfaces* session.

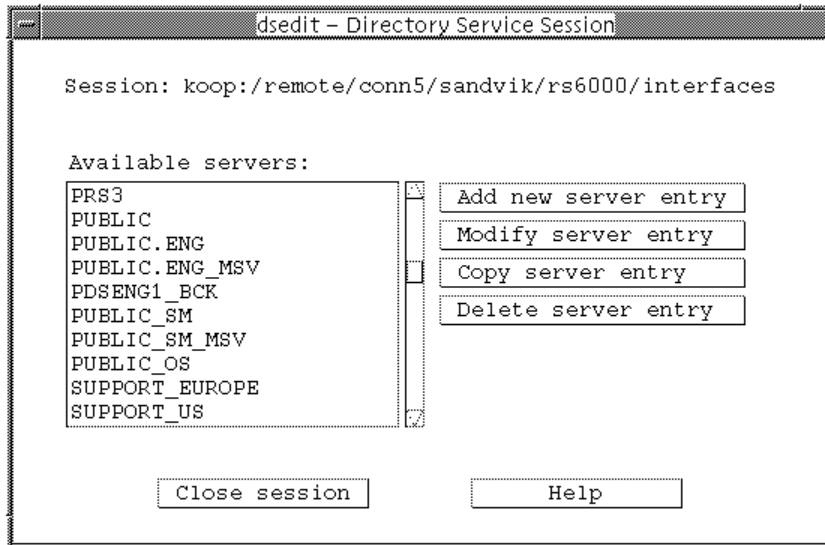


Figure 7-1: Interfaces file session window

The session window for an interfaces-file session displays the full path name of *interfaces* and lists the server entries contained in *interfaces*. The buttons to the left of the list allow you to add, modify, copy, and delete entries, as follows:

- Add new server entry – displays the Server Entry Editor window, where you specify the name and network addresses for a new server entry. See “Adding, Viewing, and Editing Server Entries” on page 7-6 for more information.
- Modify server entry – lets you view and modify the network addresses for a selected server entry. To view or modify a server entry, select the server in the list, then click Modify server entry to display the server’s attributes in the Server Entry Editor window. See “Adding, Viewing, and Editing Server Entries” on page 7-6 for more information.
- Copy server entry – lets you copy one or more entries to a DCE directory or to another *interfaces* file. Before copying server entries, select the entries to copy in the list of servers as follows:
 - To copy a single entry, click it once.

- To copy a range of consecutive entries, click the first (or last) entry in the range, then, with the Shift key pressed, click the last (or first) entry in the range.
- To select multiple, nonconsecutive entries, press and hold down the Ctrl key, then click each desired entry to select it.

After selecting entries to copy, click the Copy server entries button. A new window prompts you to choose the destination directory service to which to copy. You can copy to another *interfaces* file or (if running *dsedit_dce*) to a DCE directory, as follows:

- To copy the entries to another *interfaces* file, select Sybase Interfaces File from the list, edit the displayed file name, then click OK.
- To copy the entries to a DCE directory, select the corresponding directory service from the window, then click OK.

The Close Session button closes the session window and writes any changes to *interfaces*.

► **Note**

You must close the *interfaces* session window to apply your edits to the *interfaces*.

DCE Directory Sessions

To open a DCE directory session, you must be using *dsedit_dce*. To open a session, select the directory service name from the list, then click OK. If no directory service entries are shown except for *interfaces*, you need to configure the DCE directory driver in *libtcl.cfg*, then restart *dsedit_dce* (see “DIRECTORY Section” on page B-3 for details on directory driver configuration). Figure 7-2 shows the screen for a DCE directory session.

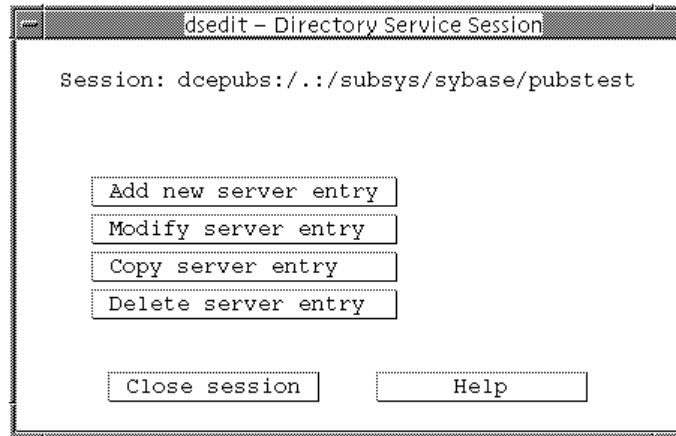


Figure 7-2: DCE directory session window

The session window for a DCE directory session displays the DIT base in the DCE directory where entries are stored. The buttons to the left of the list lets you add, modify, copy, and delete entries, as follows:

- Add new server entry – displays the Server Entry Editor window, where you specify the name and network addresses for a new server entry. See “Adding, Viewing, and Editing Server Entries” on page 7-6 for more information.
- Modify server entry – lets you view and modify the network addresses for a selected server entry. To view or modify a server entry, click Modify server entry, then enter the name of the server. `dsedit_dce` displays the server’s attributes in the Server Entry Editor window. See “Adding, Viewing, and Editing Server Entries” on page 7-6 for more information.
- Copy server entry – lets you copy one or more entries to a DCE directory or to another *interfaces* file. To copy an entry, click the Copy server entry button, then enter the name of the entry to copy. A new window will prompt you to choose the destination directory service to which to copy. You can copy to another *interfaces* file or to another DCE directory, as follows:
 - To copy the entry to another *interfaces* file, select Sybase Interfaces File from the list, edit the displayed file name, then click OK.

- To copy the entry to another DCE directory, select the corresponding directory service from the window and click OK. (In order for different DCE directories to appear as available directory services, you must have multiple DCE directory driver entries in the *libtcl.cfg* file that specify different DIT base values. See “Adding a Directory Driver” on page B-7 for information on configuring directory driver entries.)

Click Close Session to close the session window.

Adding, Viewing, and Editing Server Entries

You use the Server Entry Editor window to view or edit server entries in an *interfaces* file or DCE directory. The Add New Server Entry and Modify Server Entry buttons in the Session window display the Server Entry Editor window. This window has these controls:

- Server name – If you are adding a server entry, type the name of the new server. If you are editing a server entry, you can edit the name field to rename the server (the new name must not already exist in the DCE directory or in *interfaces*).
- Available network transports – A list of the network addresses upon which the server accepts client connections. You can edit the list of addresses as follows:
 - Select Add Network Transport or Modify Network Transport to create a new address or edit an existing address. See “Adding or Editing Network Transport Addresses” on page 7-7 for details.
 - The “Delete network transport button” removes a selected network address.
 - If the server entry has multiple addresses, the “Move network transport up” and “Move network transport down” buttons allow you to rearrange the order of addresses in the list.
- OK button – Commits your changes and closes the window. Note that changes to *interfaces* are not applied until you close the session.
- Cancel button – Closes the window and discards any edits.

Adding or Editing Network Transport Addresses

The Network Transport Editor allows you to view, edit, or create the transport addresses at which a server accepts client connections. This window displays the name of the server entry for the address and allows you to configure the following items:

- Transport type – Specifies the protocol and interface for the address. Values can be tcp, tli tcp, tli spx, or spx.
- Address information – Depending on the transport type, different address components are required. The sections below discuss address formats in detail.

TCP/IP Addresses

You indicate a TCP/IP address by choosing “tcp” or “tli tcp” from the Transport type menu. In *interfaces* entries, you must use the “tli tcp” protocol for:

- All pre-10.0 clients on platforms that use “tli” formatted *interfaces* entries,
- SQL Server or Replication Server® release 11.0.x or earlier on platforms that use “tli” formatted *interfaces* entries.

For other clients and servers, use the “tcp” transport type.

The use of the “tli tcp” transport in DCE directory entries is discouraged; a DCE directory can be shared by applications running on multiple platforms, but the “tli tcp” protocol is not recognized on all platforms. Use “tcp” for TCP/IP addresses in a DCE directory entry.

The address information for a TCP/IP entry consists of a host name (or IP address) and a port number (entered as a decimal number). For “tli tcp” formatted *interfaces* entries, the host’s IP address and the port number are converted to the 16-byte hexadecimal representation required for “tli tcp” formatted *interfaces* entries.

SPX/IPX Addresses

SPX/IPX addresses allow a UNIX SQL Server to listen for connections from client applications running on a Novell network. You indicate an SPX/IPX address by choosing “tli spx” or “spx” from the Transport type menu. SPX/IPX addresses consist of the following information:

- Host address – An 8-digit hexadecimal value representing the IP address of the computer on which the server runs. Each component of the dot-separated decimal IP address format maps to one byte in the hex address format. For example, if your host's IP address is 128.15.15.14, then enter 800F0F0E as the SPX/IPX host-address value.
- Port number – The port number, expressed as a 4-digit hexadecimal number.
- Endpoint – The path for the device file that points to the SPX device driver. Defaults to `/dev/mspx` on Solaris and `/dev/nspix` on any other platform. If necessary, adjust the path so that it is correct for the machine on which the server runs. The default path is based on the platform on which you are running `dsedit`.

Troubleshooting *dsedit* or *dsedit_dce* Problems

This section lists some common problems and describes how to correct them.

dsedit or *dsedit_dce* Does Not Start

Check for the following common trouble spots:

- SYBASE environment variable is not set or points to the wrong directory.
- X11 is not configured correctly. If you are running `dsedit` or `dsedit_dce` on a remote host, make sure that X11 clients on the remote host can connect to the X11 server on your own machine. See your X11 documentation for more troubleshooting information. If X11 is not available, use `dscp` or `dscp_dce` instead of `dsedit` or `dsedit_dce`.
- You tried to run `dsedit_dce`, but the necessary DCE libraries could not be loaded. `dsedit_dce` requires shared libraries that are supplied by your DCE installation. Make sure that the directory containing the DCE libraries is in your system's shared-library search path.

DCE Does Not Display As an Available Directory Service

Check for the following common trouble spots:

- You must run `dsedit_dce` instead of `dsedit` to edit DCE entries.

- The [DIRECTORY] section in the *libtcl.cfg* file is not configured correctly. Each entry in this section configures the directory driver for the directory service that is shown in the Select Directory Service menu. See “Adding a Directory Driver” on page B-7 for instructions on how to configure the driver for a directory service.

Cannot Open DCE Directory Session

If a DCE service is displayed in the Select Directory Service menu, but you get an error when trying to open it, then check for the following cause:

- The entry in the *libtcl.cfg* [DIRECTORY] section lists the wrong Sybase driver. See “DIRECTORY Section” on page B-3 for details on the *libtcl.cfg* file syntax, and “Dynamic Linking of Drivers” on page B-2 for a description of how drivers are loaded.

Cannot Add, Modify, or Delete Server Entries

Check for the following common trouble spots:

- Permissions problems with the interfaces file or DCE directory
To edit interfaces entries, you must have write permission on the interfaces file and the Sybase installation directory. To edit DCE entries, you must be logged into DCE as a user who has write privileges on the DCE directory.
- DIT base incorrectly configured for DCE in *libtcl.cfg* file
Each directory service entry in the *libtcl.cfg* [DIRECTORY] section lists a DIT base, which is the base path under which server objects are stored for that service. Make sure that the DIT base path actually exists in the DCE directory. See “DIRECTORY Section” on page B-3 for a description of how the DIT base is defined.
- DCE system trouble
A DCE administrator can use the DCE *dcecp* tool to verify that DCE directory software is working correctly. See your DCE documentation for more information.

Part 3: Configuration Reference

A

Environment Variables

This appendix describes environment variables that contain configuration information. The following topics are included:

- Environment Variables Used for Connection A-1
- Environment Variables Used for Localization A-1
- Setting Environment Variables A-2

Environment Variables Used for Connection

Open Client/Server products use the environment variables, shown in Table A-1, during the connection process..

Table A-1: Environment variables used for connection

Variable	Value	Used By
DSLISEN	The name of the Open Server application, as listed in <i>interfaces</i> or directory service. If DSLISEN is not set, Open Server uses the default value "SYBASE."	Open Server
DSQUERY	The name of the target server, as listed in <i>interfaces</i> or directory service. If DSQUERY is not set, Open Client uses the default value "SYBASE."	Open Client
SYBASE	The location of the installation directory for Sybase products. If SYBASE is not set, Open Client/Server products default to the user's home directory.	Open Client and Open Server

Environment Variables Used for Localization

Open Client/Server products use these environment variables during localization.

- LC_ALL
- LC_COLLATE
- LC_TYPE
- LC_MESSAGE

- LC_TIME

See Appendix C, “Localization,” for reference information about the environment variables listed above.

The localization environment variables are POSIX standard environment variables and can be used by non-Sybase applications.

Some non-Sybase applications can use the same localization-related environment variable as your Open Client/Server application. Make sure that *locales.dat* lists the same locale names as are used by the environment variables of the non-Sybase applications.

Setting Environment Variables

This section gives instructions for setting environment variables in the C shell and the Bourne shell.

C Shell

In the C shell, use this command to set environment variables:

```
setenv VARIABLE value
```

For example:

```
setenv DSQUERY test
```

defines the DSQUERY environment variable as “test.”

Bourne Shell

In the Bourne shell, use this command to set environment variables:

```
VARIABLE=value
```

For example:

```
DSQUERY=test
```

defines the DSQUERY environment variable as “test.”

B Configuration Files

This appendix describes the files that Open Client/Server products use to obtain configuration information. The following topics are included:

- About Configuration Files B-1
- *libtcl.cfg* B-1
- interfaces B-10
- *ocs.cfg* File B-16

About Configuration Files

Configuration files are created during installation at a default location in the *\$\$SYBASE* directory structure. Open Client/Server products use the configuration files listed in Table B-1.

Table B-1: Names and locations for configuration files

File Name	Description	Location	For Further Information
<i>libtcl.cfg</i>	The driver configuration file contains information regarding directory, security, and network drivers and any required initialization information.	<i>\$\$SYBASE/config</i>	page B-1
<i>interfaces</i>	<i>interfaces</i> file contains connection and security information for each server listed in the file. It is also used as a backup to the <i>libtcl.cfg</i> file.	<i>\$\$SYBASE</i>	page B-10
<i>objectid.dat</i>	The object identifiers file maps global object identifiers to local names for character set, collating sequence, and security mechanisms.	<i>\$\$SYBASE/config</i>	Appendix C, "Localization"
<i>ocs.cfg</i>	The runtime configuration file allows you to change certain values at runtime.	<i>\$\$SYBASE/config</i>	page B-16

libtcl.cfg

libtcl.cfg, the driver configuration file, contains information about three types of drivers used by Open Client/Server products:

- Directory drivers
- Security drivers
- Network (Net-Library) drivers

A driver is a Sybase library which provides Open Client/Server software with a generic interface to an external service provider. This allows Open Client and Open Server to easily support multiple service providers. For example, Open Client can use the DCE directory driver to communicate with the DCE directory service.

Dynamic Linking of Drivers

Client-Library and Server Library support dynamic loading of Net-Library, directory, and security drivers. This allows you to change a driver that an application uses and to use features as they become available at your site, without relinking the application.

\$\$SYBASE/config/libtcl.cfg configures Net-Library, directory, and security drivers. This file maps symbolic strings to the appropriate driver and any required initialization information.

Client-Library or Server-Library applications, including Sybase utility programs such as *dscp_dce*, locate the appropriate drivers specified in *libtcl.cfg* by following these steps:

1. If the driver file name in *libtcl.cfg* has path components (contains a slash), that path is used. Otherwise, the search continues to step 2.
2. Depending on your platform, the directories specified by these environment variables are searched:
 - Sun Solaris – LD_LIBRARY_PATH
 - HP/UX – SH_LIB_PATH
 - IBM RS6000 – LIBPATHIf the driver is not located, the search continues to step 3.
3. The path *\$\$SYBASE/lib* is used (or *\$\$SYBASE/devlib* for applications built with debug-mode libraries).

How *libtcl.cfg* Is Used

Open Client and Open Server read the *libtcl.cfg* file when loading a network, directory, or security driver.

An entry in *libtcl.cfg* provides Open Client/Server products with the name of the driver and its initialization information.

Location of *libtcl.cfg*

libtcl.cfg is located in the *\$\$SYBASE/config* directory.

Layout of *libtcl.cfg*

libtcl.cfg is divided into three sections, one for each type of driver. The sections are titled:

- [DIRECTORY]
- [SECURITY]
- [DRIVERS]

► **Note**

The sections do not have to be in a specific order.

DIRECTORY Section

The DIRECTORY section lists directory drivers. The syntax for a directory driver entry is:

```
provider=driver init-string
```

where:

- *provider* is a local name of the directory service. You can name this element anything as long as it contains only letters, numbers, and underscores and has a maximum of 64 characters.
- *driver* is the name of the driver. The default location of all drivers is in *\$\$SYBASE/lib*. The DCE directory driver is platform dependent:
 - HP/UX – *libddce.sl*
 - IBM RS/6000 – *libddce.so.a*
 - Sun Solaris – *libddce.so*
- *init-string* is an initialization string for the driver. The value for *init-string* varies by driver.

For the DCE driver, *init-string* specifies the DIT base. DIT base is the location where DCE begins its search for the server entry. The syntax for *init-string* is:

```
ditbase=././location
```

If *init-string* is not specified, the default DIT base is *././subsys/sybase/dataservers*.

Run the DCE *dcecp* utility and use the *create directory* command to create the DIT base location in the DCE directory structure where server entries can be located. See your DCE documentation for more information on the *dcecp* utility.

The following DIRECTORY section shows an entry for a Sun Solaris DCE driver:

```
[DIRECTORY]
dce=libddce.so ditbase=././subsys/sybase/server
```

SECURITY Section

The SECURITY section lists security drivers. The syntax for a security driver entry is:

```
provider=driver init-string
```

where:

- *provider* is the local name for the security mechanism. The local name of the security mechanism is listed in the object identifiers file, *\$\$YBASE/config/objectid.dat*.

See “objectid.dat” on page C-12 for information about *objectid.dat*.

The default local name for the DCE security mechanism is “dce.” The default local name for the CyberSAFE Kerberos security mechanism is “csfkrb5”. If you use a local mechanism name other than the default, you must add an alias for the name in the object identifiers file, after the default name (see “objectid.dat File Fragment” on page C-12 for an example).

- *driver* is the name of the driver. The default location of all drivers is in *\$\$YBASE/lib*.

Table B-2 lists the supported security drivers for each platform.

Table B-2: Supported security drivers

Platform	Security Type	Security Driver	Service Compatibilities
Solaris 2.x	DCE	<i>libsdce.so</i>	Operating-system provider must be compatible with OSF DCE 1.1
	CyberSAFE Kerberos	<i>libskrb.so</i>	CyberSafe Application Security Toolkit 1.05
IBM RS/6000	DCE	<i>libsdce.so.a</i>	Operating-system provider must be compatible with OSF DCE 1.1
HP-UX	DCE	<i>libsdce.sl</i>	Operating-system provider must be compatible with OSF DCE 1.1
	CyberSAFE Kerberos	<i>libskrb.sl</i>	CyberSafe Application Security Toolkit 1.05

- *init-string* is an initialization string for the driver. The value for *init-string* varies by driver.

For the DCE driver, the syntax for *init-string* is:

```
secbase=../../cell_name
```

Where *cell_name* is the name of your DCE cell.

For the CyberSAFE Kerberos driver, the syntax for *init-string* is:

```
secbase=@realm
```

Where *realm* is the default CyberSAFE Kerberos realm name.

The following SECURITY sections show entries for DCE and CyberSAFE Kerberos drivers on Sun Solaris:

- DCE

```
[SECURITY]
dce=libsdce.so secbase=../../dsatestcell
```

- CyberSAFE Kerberos

```
[SECURITY]
csfkrb5=libskrb.so secbase=@csfrealm
```

DRIVERS Section

The DRIVERS section lists network drivers. The syntax for a directory driver entry is:

```
driver=protocol description
```

where:

- *driver* is the name of the driver. The default location of all drivers is in *\$\$SYBASE/lib*.

The network transport drivers are:

- *libtli.so* for tli – Sun Solaris
- *libinsck.sl* for tcp – HP 9000, RS/6000
- *libinsck.so.a* for tcp – RS/6000

- *protocol* is the name of the network protocol. The valid values are:
 - “tcp” for TCP/IP
 - “tli” for TLI

This element must match the protocol of the driver.

- *description* is a description of the entry. This element is optional.

The following DRIVERS section shows an entry for a tli driver:

```
[DRIVERS]
libtli.so=tli The tli driver
```

► **Note**

If you do not specify a driver in the DRIVERS section, Open Client/Server products use a platform-specific default driver. See Table B-3.

Network Drivers

Table B-3 shows the supported network drivers and the default transport-driver mappings for UNIX platforms.

Table B-3: Default transport-type driver mappings

Platform	Transport Type	Net-Library Driver
Solaris 2.x	tcp or tli	<i>libtli.so</i> (for applications linked with non-threaded versions of the Sybase libraries) <i>libtli_r.so</i> (for applications linked with the threaded Sybase libraries)
IBM RS/6000	tcp	<i>libinsck.so.a</i> (for applications linked with non-threaded versions of the Sybase libraries) <i>libinsck_r.so.a</i> (for applications linked with the threaded Sybase libraries)
HP-UX	tcp	<i>libinsck.sl</i> (for applications linked with non-threaded versions of the Sybase libraries) <i>libinsck_r.sl</i> (for applications linked with the threaded Sybase libraries)

The 11.1 libraries for Solaris 2.x recognize the tli transport type in order to provide backward compatibility with existing *interfaces* entries.

Adding a Directory Driver

To add a directory driver to *libtcl.cfg*:

1. Choose a value for *provider*.
provider can have any value.
2. Determine the value of *driver*.
driver is the name of the driver. This value varies by platform:
 - For HP/UX, use *libddce.sl*
 - For IBM RS/6000, use *libddce.so.a*
 - For Sun Solaris, use *libddce.so*

► Note

For HP/UX, IBM RS/6000, and Sun Solaris, the operating-system provider must be compatible with OSF DCE 1.1.

3. Determine the value of the DIT base.

The DCE DIT base is the location where DCE begins its search for the server entry.

4. Ensure that the DIT base path exists in the DCE directory.

The DCE administrator may need to perform this task. Run `dcecp` and use the `directory list` command to determine whether the DIT base location exists. If necessary, use the `dcecp` utility's `directory create` command to create the DIT base location. See your DCE documentation for more information on `dcecp`.

5. Go to the DIRECTORY section and add an entry using the following format:

```
provider=driver ditbase=Value
```

For example:

```
dce=libddce.so ditbase=././subsys/sybase/testserver
```

► Note

To make an entry the default directory driver, add it as the first entry in the DIRECTORY section.

You can add two or more DCE driver entries that use different DIT bases. Multiple driver entries are useful when you want to use the `dscp_dce` or `dsedit_dce` tools to view and modify entries that are in different locations in the DCE directory. For example, you might add the entries below:

```
[DIRECTORY]
dce=libddce.so ditbase=././subsys/sybase/dataserver
dctest=libddce.so ditbase=././subsys/sybase/testserver
```

Adding a Security Driver

To add a security driver to *libtcl.cfg*:

1. Determine the value of *provider*:

provider is the local name of the security mechanism, as listed in the object identifiers file, *\$\$YBASE/config/objectid.dat*. The default local name for the DCE security mechanism is “dce.”

See “objectid.dat” on page C-12 for information about *objectid.dat*.

2. Determine the value of *driver*:

driver is the name of the driver, which varies by platform and security mechanism (Table B-2 on page B-5 lists driver names by security mechanism and platform).

3. Determine the value of *init-string*:

For the DCE driver, *init-string* has the form:

```
secbase=.../cellname
```

where *cellname* is the name of your DCE cell.

For the CyberSAFE Kerberos driver, *init-string* has the form:

```
secbase=@realmname
```

where *realmname* is the default realm name for unqualified CyberSAFE user names.

4. Go to the SECURITY section and add an entry using the following format:

```
provider=driver init-string
```

For example:

```
dce=libsdc.so secbase=./:/sec/principal
```

Adding a Network Driver

To add a network driver to *libtcl.cfg*:

1. Determine the value of *driver*:

driver is the name of the driver. The drivers are:

- *libinsck.sl* for TCP/IP on HP 9000
- *libinsck.so.a* for TCP/IP on RS/6000
- *libtli.so* for TLI

The driver your Open Client/Server product uses varies by platform. The following table shows which platforms are compatible with each driver:

Platform	TCP/IP Driver <i>libinsck.sl</i> or <i>libinsck.so.a</i>	TLI Driver <i>libtli.so</i>
HP 9000 HP-UX	x	
IBM RS/6000	x	
Sun Solaris (SPARC)		x

2. Determine the value of *protocol*.

protocol must match the protocol of the driver you selected in step 1. The valid values are:

- “tcp” for TCP/IP
- “tli” for TLI

3. Choose a value for *description*.

description can have any value and is optional.

4. Go to the DRIVERS section and add an entry using the following format:

```
driver=protocol description
```

For example:

```
libtli.so=tli The tli driver
```

► **Note**

If you do not specify a driver in the DRIVERS section, Open Client/Server products use a platform-specific default driver. See Table B-3 on page B-7.

interfaces

interfaces contains information about the network locations of servers.

How It Is Used

Open Client and Open Server use *interfaces* as a limited-function directory service. *interfaces* also serves as a default if an external directory service fails.

- Open Client uses the network information provided by the **query** line of an *interfaces* entry to connect to the server.
- Open Server uses the network information provided by the **master** line of an *interfaces* entry to listen for client connection requests.

Location of *interfaces*

interfaces is created during installation as *SSYBASE/interfaces*. Open Client/Server products look for *interfaces* in *SSYBASE*.

An application can look for *interfaces* in a location other than the default location. See *ct_config* in the *Open Client Client-Library/C Reference Manual* and *srv_props* in the *Open Server Server-Library/C Reference Manual* for more information.

interfaces Entries

Open Client and Open Server release 11.1 use a standard format for *interfaces* entries.

► Note

Past releases of Open Client/Server and the current release of SQL Server require a hexadecimal entry format for tli-based platforms. This section defines the standard format and discusses when to use the old tli format.

Standard Format

An *interfaces* entry has the form:

```
# put comments here<newline>
SERVERNAME[<tab>retry_count<tab>retry_delay]<newline>
<tab>{master|query} protocol network host port<newline>
<tab>[secmech mechanism1,..., mechanismn]<newline>
<blank line>
```

where:

- *SERVERNAME* is an alias by which Open Client and Open Server recognize which *interfaces* entry to read. *SERVERNAME* must begin with a letter (ASCII a-z, A-Z), contain letters, numbers, and underscores only, and have a maximum of 11 characters.
- *retry_count* determines the number of times a client tries to connect to a server after an initial failure to connect. *retry_count* is optional.
- *retry_delay* determines the time interval between connection attempts. *retry_delay* is optional.
- “master | query” specifies the type of connection:
 - “master” specifies a master line, which is used by server applications to listen for client queries.
 - “query” specifies a query line, which is used by client applications to find servers.

The master line and the query line of an *interfaces* entry contain identical information. *dscp* creates both types of lines for each entry. The resulting entry can be used by both clients and servers.

- *protocol* is the name of the network protocol. The valid values are:
 - “tcp” for TCP/IP – All UNIX platforms
 - “spx” for IPX/SPX – UnixWare
 - “decnet” for DECnet – Digital UNIX

► **Note**

SQL Server requires that *interfaces* entries for transport layer interface (tli) protocols use the hexadecimal format. See “Transport Layer Interface Format” on page B-14 for more information.

- *network* is a descriptor of the network.

Open Client and Open Server do not currently use *network*; it is a placeholder should Sybase need to define this information in the future.
- *host* is the network name of the node, or machine, that the server is running on. The maximum number of characters for *host* depends on the protocol specified in the entry:
 - For TCP/IP the maximum is 32.
 - For DECnet the maximum is 6.

Use the `/bin/hostname` command to determine the network name of the machine you are logged into.

- *port* is the port used by the server to receive queries. The TCP/IP and DECnet protocols specify this element differently:
 - TCP/IP: Valid port numbers range from 1025 to 65535. However, the numbers 1025 to 7009, 9535 and 17007 are registered port numbers and may already be in use on your system.
 - DECnet: Valid object numbers range from 128 to 253. Object names are also valid.

Use the `netstat` command to check which port numbers are in use.

- “secmech” is the identifier used to list the security mechanisms that a server supports. The secmech line is optional.
- *mechanism1, ..., mechanismn* are the security mechanisms that a server supports. You can specify multiple security mechanisms by using a “,” separator.

A security mechanism is listed as its object identifier. An object identifier is globally unique series of numbers which maps to the local name for a security mechanism in the global object identifiers file.

See “objectid.dat” on page C-12 for more information about object identifiers.

Figure B-1 shows a fragment of an *interfaces* file with entries for servers BETA, TEST, PRODUCTION, and a list of their components.

```

#
BETA
query tcp ether violet 1025
master tcp ether violet 1025
secmech 1.3.6.1.4.1.897.4.6.1
#
TEST
query tcp ether plum 1050
master tcp ether plum 1050
secmech 1.3.6.1.4.1.897.4.6.1
#
PRODUCTION _____ SERVERNAME
|_____ protocol
|_____ network
query tcp ether mauve 1060 _____ port
|_____ host
master tcp ether mauve 1060
secmech 1.3.6.1.4.1.897.4.6.1 _____ mechanism 1
|_____ object identifier(OID)

```

Figure B-1: *interfaces* File Example

Transport Layer Interface Format

While Open Client and Open Server release 11.1 use a standard format for *interfaces* entries, they also recognize entries which use the tli hexadecimal format.

SQL Server release 11.0 continues to require that entries for tli-based platforms use the hexadecimal format.

◆ **WARNING!**

You must use the tli hexadecimal format if you make an entry for a tli-based SQL Server.

dscp allows you to specify the tli format by entering:

```
tli protocol
```

as the transport type. See “Adding and Modifying Server Entries” on page 6-5 for more information.

See the *SQL Server Configuration Guide* for more information about the tli hexadecimal format.

Editing *interfaces*

Edit *interfaces* with `dscp` or an operating system editor, such as `vi`.

Using `dscp` to edit an *interfaces* file makes the process easier because it correctly formats the address string that you enter. See Chapter 6, “Using `dscp` and `dscp_dce`,” for complete instructions about editing an *interfaces* file with `dscp`.

Standby Server Addressing

You can set up your *interfaces* file to allow for **standby server addressing**. Standby server addressing allows Open Client to connect with an alternate server if the first connection attempt fails.

For example, the *interfaces* entry below directs the application to the server at port number 1025 on the machine *violet*. If this server is not available, the connection fails.

```
#
BETA
  query tcp hp-ether violet 1025
  master tcp hp-ether violet 1025
  secmech 1.3.6.1.4.1.897.4.6.1
```

However, if the BETA entry has multiple *query* lines, Open Client automatically attempts to connect to the next server listed when the first connection attempt fails. Such an *interfaces* entry might look like the following example:

```
#
BETA
  query tcp hp-ether violet 1025
  query tcp hp-ether plum 1050
  query tcp hp-ether mauve 1060
  master tcp hp-ether violet 1025
  secmech 1.3.6.1.4.1.897.4.6.1
```

► **Note**

The *SERVERNAME* element of an *interfaces* entry is an **alias** and does not uniquely identify the actual server. The host and port elements uniquely identify the server.

In the example above, if Open Client fails to connect to violet at port 1025, Open Client attempts to connect to the server listed in the next query line, plum at port 1050, and so on.

Any number of alternate servers may be listed under a server's *interfaces* entry, but each alternate server must be listed in the same *interfaces* file.

***ocs.cfg* File**

ocs.cfg, the runtime configuration file is used by Client-Library applications to set:

- Property values
- Server option values
- Server capabilities
- Debugging options

By using *ocs.cfg*, applications eliminate the need to call routines to set values. A benefit of using *ocs.cfg* is that the application's settings can be changed without recompiling the code.

Client-Library does not read *ocs.cfg* by default. The application must set properties to enable Client-Library to use this file.

See "Using the Open Client/Server Run-Time Configuration File" in the *Open Client Client-Library Reference Manual* for information about the file syntax and the properties that can be set in the file.

C

Localization

Localization is the process of initializing an application so that it executes using a specific language and related cultural conventions.

This appendix discusses localization and localization files from a system configuration perspective. For a discussion of programming issues related to localization, see the *Open Client/Server International Developer's Guide*.

The following topics are included:

- Overview of the Localization Process C-1
- Localization Files C-3
- The locales Directory C-5
- The charsets Directory C-8
- The config Directory C-12

Overview of the Localization Process

Open Client/Server applications can localize in two different ways:

- Using initial localization values
- Using initial localization values and custom localization values

All Open Client/Server applications use initial localization values, which are determined at run time.

Additionally, an Open Client/Server application can use custom localization values if there is the need to localize at a specific point during the application's execution. Custom localization values override the initial localization values that are set up at run time.

Environment Variables Used During Localization

Open Client and Open Server use environment variables to determine which locale name to look for in *locales.dat*.

Open Client and Open Server always search for the following environment variables:

- LC_ALL
- LANG, if LC_ALL is not set

When setting up custom localization values, Open Client and Open Server may also search for one or more of the environment variables shown in Table C-1.

Table C-1: Environment variables used for localization

Environment Variable	Description	Used During
LC_ALL	Language, character set, and collating sequence to use for messages, datatype conversions, and sorting.	Initial localization, Custom localization
LANG	Language, character set, and collating sequence to use for messages, datatype conversions, and sorting. Open Client/Server products search for LANG if they cannot find LC_ALL.	Initial localization
LC_COLLATE	Collating sequence (sort order) to use when sorting and comparing character data.	Custom localization
LC_CTYPE	Character set to use for datatype conversions.	Custom localization
LC_MESSAGE	Language to use for messages.	Custom localization
LC_TIME	Date and time data representation to use for a datetime string, such as date and time formats, names in the native language, and month and day abbreviations.	Custom localization

See the *Open Client/Server International Developer's Guide* for more information about what environment variables an application uses during custom localization.

Before running a localized application, check the following:

- Make sure *locales.dat* contains an entry which reflects the localization values the application will use. If it does not, add an appropriate entry.
- Make sure that the localization files that your application will use are installed:
 - Localized message files are located in the `$$SYBASE/locales/message` directory.
 - Collating sequence files are located in the `$$SYBASE/charsets` directory.

Open Client/Server products come with the localization files to support one language and one or more character sets and sort orders.

Localization Files

At runtime, Open Client/Server applications load localization information from external files. Three directories in the `$$SYBASE` directory contain these files:

- The *locales* directory contains:
 - The locales file, *locales.dat*, which maps locale names to languages, character sets, and collating sequences
 - The *message* subdirectory, which contains localized error messages for all products, organized by language name.
 - *language_name* subdirectories, which are included to provide compatibility with previous releases of Open Client/Server software. These directories contain localized message files organized by character set.
- The *charsets* directory contains a subdirectory for each supported character set. Each subdirectory contains sort and conversion files for the character set.
- The *config* directory contains:
 - The global object identifiers file, *objectid.dat*, which maps global names for objects such as character sets and languages to local platform-specific names.
 - The mnemonics file, *mnemonic.dat*, which provides mnemonic strings for replacing source Unicode, if necessary.

All Open Client/Server products include files to support at least one language and one or more character sets and collating sequences.

During installation, these files are loaded into the `$$SYBASE` directory structure in the appropriate locations.

Figure C-1 illustrates how localization files appear in the Sybase release directory structure. Shading indicates directories included for compatibility with previous releases:

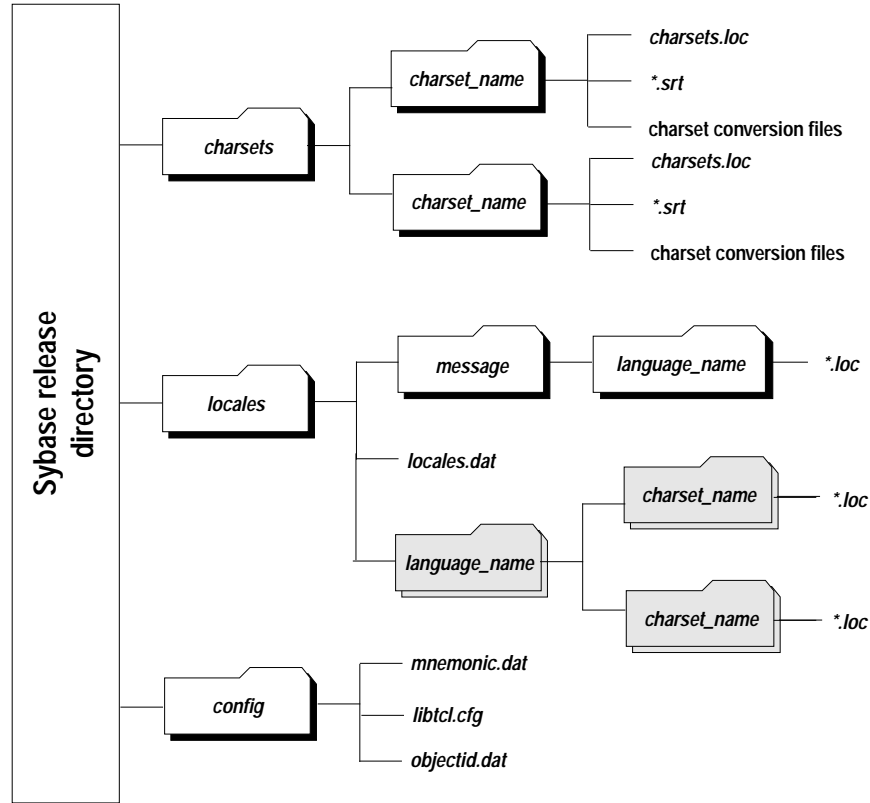


Figure C-1: Sybase release directory structure

When configuring an Open Client or Open Server application, you must make sure that the above directories contain the correct files for your site and application.

The *locales* Directory

The *locales* directory contains files that your application uses to load localization information. It also contains language-specific message files.

locales.dat

locales.dat, the locales file, provides platform-specific locale information in a Sybase proprietary format. This file associates locale names with languages, character sets and collating sequences.

How It Is Used

Open Client/Server applications use *locales.dat* to determine what localization information to load. *locales.dat* directs Open Client/Server applications to localization information, but it does not contain actual localized messages or character set information.

Location of *locales.dat*

locales.dat is located in the *\$\$SYBASE/locales* directory.

See “Localization Files” on page C-3 for a diagram of the *\$\$SYBASE/locales* directory structure.

locales.dat Sections and Entries

locales.dat contains platform-specific sections, each of which contains predefined locale definition entries. These entries vary by platform, but all sections include an entry defining a “default” locale.

Locale definition entries have the form:

```
locale = locale_name, language_name, charset_name  
[ , sortorder_name ]
```

where:

- *locale_name* is the name of the locale definition. The default values for *locale_name* are vendor-specified and based on POSIX terminology. Comments at the end of *locales.dat* list POSIX values for locale names.
- , (comma) is the list separator character for the file.

- *language_name* is the subdirectory name by which Sybase products recognize the language.
- *charset_name* is the subdirectory name by which Sybase products recognize the character set.
- *sortorder_name* is the file name by which Sybase products recognize the collating sequence (optional).

The following *locales.dat* file entry specifies a French locale. Because no sort order is specified, the default sort order “binary” is used with this locale:

```
locale = fr.FR.88591, french, iso_1
```

locales.dat File Fragment

The following fragment illustrates a platform-specific section in *locales.dat*:

```
[aix]
locale = C, us_english, iso_1
locale = En_US, us_english, iso_1
locale = en_US, us_english, iso_1
locale = default, us_english, iso_1
locale = japanese.sjis, japanese, sjis
locale = japanese, japanese, eucjis
locale = us_english.utf8, us_english, utf8
```

Editing *locales.dat*

If the predefined entries in *locales.dat* do not meet your needs, edit the file with an operating system editor such as vi.

◆ **WARNING!**

Before you edit, make a copy of the original *locales.dat*. The copy will help you solve any problems with the edited version. Also, review the entries for your platform to see if an entry already exists.

Edit *locales.dat* to:

- Change the “default” locale definition.
- Add a locale definition.
- Match a locale name used by non-Sybase software. For example, the Sybase predefined locale name is “fr”:

```
locale = fr, french, iso_1
```

If a non-Sybase application requires a value of “french” for the LC_ALL environment variable, change the locale name to:

```
locale = french, french, iso_1
```

To add a new entry to *locales.dat* or to change an existing entry:

1. Choose a value for *locale_name*.

locale_name can have any value.

2. Determine the value for *language_name*.

When a Sybase language module is installed, a subdirectory for the language is created in the *locales/message* directory of the Sybase directory tree. *language_name* must correspond to this subdirectory's name.

3. Determine the value for *charset_name*.

When a Sybase language module is installed, subdirectories for each supported character set are created in the *charsets* directory of the Sybase directory tree. *charset_name* must correspond to one of these subdirectory names.

4. Determine the value for *sortorder_name* (if you want a sort order other than binary).

The *charsets/charset_name* subdirectory contains the sort order (*.srt) files for the character set. *sortorder_name* must correspond to one of these file names (without the .srt).

5. In the appropriate platform-specific section of the *locales.dat* file, type in or change the appropriate entry.

After you make the change:

- Update localization environment variables (LC_ALL, LC_CTYPE, LC_MESSAGE, LC_TIME, LANG) as appropriate.
- If you have added a new locale name and you want existing applications to use this new name in *cs_locale* calls, edit and recompile the applications as appropriate.

It is not necessary to delete entries from *locales.dat*, even if applications no longer use them. If you decide to delete an entry, make sure no application uses it.

Localized Message Files

◆ **WARNING!**

Do not edit localized message files.

Localized message files contain product messages in a particular language. These message files (the *.loc files in the *locales/message/language_name* directories) enable Open Client/Server applications to generate messages in a variety of languages.

All Open Client/Server products include English (us_english) message files. Your products may also include files to support additional languages.

If you purchase and install a new language module, the installation process adds a *language_name* subdirectory containing message files in the new language.

Message file names sometimes vary by platform, but most resemble the following names:

- *cslib.loc* – CS-Library messages
- *ctlib.loc* – Client-Library messages
- *oslib.loc* – Server-Library messages
- *blklib.loc* – Bulk Library messages
- *bcp.loc* – Bulk Copy messages
- *esql.loc* – Embedded SQL messages

All Open Client/Server message files use the Unicode ISO 10646 UTF-8 character set.

Open Client/Server products convert messages from UTF-8 to other character sets as needed.

The *charsets* Directory

The *charsets* directory contains conversion and collating sequence files for each supported character set.

Conversion Configuration Files

The conversion configuration file for a character set contains information on how the conversion process should proceed.

How Conversion Configuration Files Are Used

When clients and servers use different character sets, conversion between the character sets is necessary. Open Client/Server products include files to support conversions for each character set.

The conversion configuration file for a character set specifies what mode to use for the conversion and what replacement character to use for unmappable characters.

Table C-2 describes conversion modes.

Table C-2: Coded character set conversion modes

Mode	Description
MATCH Shipped files contain this value.	The conversion process converts matching source and destination values. If the code for a source character is illegal or unmappable, the conversion process uses the destination replacement character defined in the destination character set's conversion configuration file.
BESTGUESS	The conversion process converts matching and best-guess source and destination values. If the code for a source character is illegal or unmappable, the conversion process uses the destination replacement character defined in the destination character set's conversion configuration file.
MNEMONIC	Converts matching source and destination values. If there is no match for a source value, the conversion process uses a Unicode mnemonic string as the destination value. If there is no suitable mnemonic string, the conversion process uses a Unicode hexadecimal string as the destination value. If the code for a source character is illegal, the conversion process uses the destination replacement character defined in the destination character set's conversion configuration file.

The *Open Client/Server International Developer's Guide* contains a detailed description of the character set conversion process.

Location of the Conversion Configuration Files

Each character set has a conversion configuration file. This file is located at *\$\$SYBASE/charsets/charset_name/charset_name.cfg*.

See “Localization Files” on page C-3 for a diagram of the *\$\$SYBASE/charsets* directory structure.

Conversion Configuration File Entries

The conversion section contains entries that describe how conversion to a particular character set should take place.

Conversion section entries can indicate either table-driven or algorithm-driven conversion.

Table-driven entries have the form:

```
[conversion]
  convertto = dest_charset, table, mode, replacement_char
```

where:

- *dest_charset* is the name of the destination character set.
- , (comma) is the list separator character for the file.
- *table* is a keyword that indicates that the conversion is table-driven.
- *mode* is the conversion mode to use. It applies to table-driver conversions only. The valid values are:
 - MATCH
 - BESTGUESS
 - MNEMONIC

See Table C-2 for a complete description of each mode.

- *replacement_char* is a hexadecimal (without 0x prefix) encoding of the destination replacement character to use during MATCH and BESTGUESS mode conversions.

Algorithm-driven entries have the form:

```
[conversion]
  convertto = dest_charset, sys_algorithm, multiplier
```

where:

- *dest_charset* is the name of the destination character set.
- , (comma) is the list separator character for the file.

- `sys_algorithm` is a keyword that indicates that the conversion uses a standard Open Client/Server conversion algorithm.
- `multiplier` is an integer value representing the conversion multiplier for the conversion. This value indicates the maximum amount that strings may increase in length during conversion.

Conversion Configuration File Example

The following is an example of a conversion configuration file:

```
; Conversion Configuration File for iso_1 charset.
[conversion]
convertto = utf8, table, MATCH, 3F
convertto = cp850, sys-algorithm, 1
convertto = cp437, sys-algorithm, 1
convertto = roman8, sys-algorithm, 1
convertto = mac, sys-algorithm, 1
```

Collating Sequence Files

◆ **WARNING!**

Do not edit collating files.

The order in which a system sorts characters is called its **collating sequence** or **sort order**.

Open Client/Server products include files to support a variety of collating sequences. These files can vary by platform but generally include the following:

- `binary.srt`
- `dictionary.srt`
- `noaccents.srt`
- `nocase.srt`
- `nocasepref.srt`

Collating sequences are specified in `locales.dat` entries. If a `locales.dat` entry does not specify a collating sequence, then a binary sort order is used with the locale.

For more information about collating sequences, see the *Open Client/Server International Developer's Guide*.

The *config* Directory

The *config* directory contains:

- The global object identifiers file (*objectid.dat*).
- The mnemonics file (*mnemonic.dat*).

objectid.dat

objectid.dat, the global object identifiers file, associates a unique global object identifier with the local name of an object.

An object identifier is a series of non-negative integer values separated by a dot. An object identifier is based on a naming tree defined by the international standards bodies CCITT and ISO.

Location of *objectid.dat*

objectid.dat is located in the *\$\$SYBASE/config* directory.

objectid.dat Sections and Entries

objectid.dat contains a section for each class of object.

Object class entries have the form:

```
[Object Class]
    object_identifier local_name1, ..., local_namen
```

where:

- *Object Class* is the section identifier.
- *object_identifier* is the globally unique object identifier.
- *local_name1, ..., local_namen* are the local names associated with the object identifier, separated by a comma.

objectid.dat File Fragment

The following fragment illustrates sections in *objectid.dat*:

```
[charset]
    1.3.6.1.4.1.897.4.9.1.1 = iso_1
    1.3.6.1.4.1.897.4.9.1.2 = cp850
    1.3.6.1.4.1.897.4.9.1.3 = cp437
    1.3.6.1.4.1.897.4.9.1.4 = roman8
    1.3.6.1.4.1.897.4.9.1.5 = mac
```

```
[collate]
1.3.6.1.4.1.897.4.9.3.50 = binary
1.3.6.1.4.1.897.4.9.3.51 = dictionary
1.3.6.1.4.1.897.4.9.3.52 = nocase
1.3.6.1.4.1.897.4.9.3.53 = nocasepref
1.3.6.1.4.1.897.4.9.3.54 = noaccents

[secmech]
1.3.6.1.4.1.897.4.6.1 = dce
1.3.6.1.4.1.897.4.6.2 = nds
1.3.6.1.4.1.897.4.6.3 = NTLM
1.3.6.1.4.1.897.4.6.6 = csfkrb5
```

Editing *objectid.dat*

Edit *objectid.dat* if you change the local name of an object. Edit the file with an operating system editor, such as *vi*.

mnemonic.dat

mnemonic.dat, the mnemonics file, contains POSIX mnemonic strings that can be used to replace unmappable source characters, if necessary, during character-set conversion.

mnemonic.dat contains only UCS-2 <-> mnemonic string conversions. Each Unicode mnemonic in the shipped *mnemonic.dat* file is a string of XPG4 characters representing a Unicode character.

mnemonic.dat works by associating POSIX mnemonic strings with Unicode UCS-2 character encodings. Because the mnemonic strings use characters from the XPG4 Portable Character Set, the strings are suitable for use in any destination character set.

How *mnemonic.dat* Is Used

mnemonic.dat is used only if the conversion configuration file (*charset.cfg*) for a destination character set specifies a mode of "Mnemonic." If this is the case, then at conversion time *mnemonic.dat* is used as follows:

1. If a source character is found to be unmappable in the destination character set, Sybase software converts the source character to Unicode UCS-2.
2. Sybase looks up the UCS-2 encoding in *mnemonic.dat* and uses the mnemonic string associated with it in the destination data stream.
3. If *mnemonic.dat* does not contain a suitable string, a Unicode UCS-2 hexadecimal string is used in the destination data stream.

See the *Open Client/Server International Developer's Guide* for a detailed description on the character-set conversion process.

Location of *mnemonic.dat*

mnemonic.dat is located in the *\$\$SYBASE/config* directory.

See "Localization Files" on page C-3 for a diagram of the *\$\$SYBASE/charsets* directory structure.

***mnemonic.dat* Entries**

mnemonic.dat contains entries that associate UCS-2 encodings with mnemonic strings.

Mnemonics section entries have the form:

```
mnem = <mnem_string> <UCS-2_encoding>    comment
```

where:

- < is ignored.
- *mnem_string* is the string of XPG4 characters representing the mnemonic string
- > is the list separator character for the file.
- *UCS-2_encoding* is the UCS-2 encoding for a character.
- *comment* is a comment string.

Mnemonics section entries look somewhat different from entries in other Sybase localization files. This is because Sybase uses standard POSIX definitions in *mnemonic.dat*.

***mnemonic.dat* Example**

The following is an example mnemonics file fragment:

```
[file format]
  version = 11.0
  escape = /
  list_separator = >

[copyright]
  copyright = "Copyright ... ."

[mnemonics]
  mnem = <NU> <U0000> NULL (NUL)
  mnem = <SH> <U0001> START OF HEADINGS(SOH)
  mnem = <SX> <U0002> START OF TEXT (STX)
  mnem = <EX> <U0003> END OF TEXT (ETX)
  mnem = <ET> <U0004> END OF TRANSMISSION (EOT)
  mnem = <EQ> <U0005> ENQUIRY (ENQ)
  mnem = <AK> <U0006> ACKNOWLEDGE (ACK)
  mnem = <BL> <U0007> BELL (BEL)
  mnem = <BS> <U0008> BACKSPACE (BS)
  .
  .
  .
```

Adding Strings to *mnemonic.dat*

mnemonic.dat that Sybase ships does not contain strings for all characters in all character sets.

If *mnemonic.dat* does not contain a string that you need, you can insert the string using an operating system editor, such as *vi*.

The ftp site unicode.org has information on new Unicode mnemonic strings as well as updates to existing strings.

D

DCE Security Services

This appendix lists the security services supported by the DCE security driver and summarizes configuration tasks that are required in order to use the DCE security driver.

For an overview of the Open Client and Open Server security services architecture, see Chapter 5, “Using Security Services.”

Supported Security Services

The DCE security mechanism provides these security services:

- Network authentication
- Mutual authentication
- Data integrity
- Data confidentiality
- Replay detection
- Out of sequence detection
- Credential delegation
- Data origination
- Channel binding

See the *Open Client Client-Library/C Reference Manual* for a description of these security services.

Configuring DCE for Use with Open Client and Open Server

This section summarizes administration tasks that are required to use DCE security with Open Client/Server applications running within your DCE cell.

Some DCE administration tasks described here require you to use the DCE `dcecp` tool. See your DCE documentation for a description of this tool.

Open Server Applications and DCE Security

You can run a custom Open Server application or the Security Guardian server in your DCE cell.

In order for the server and its clients to communicate over the network, you must perform the normal configuration steps described in Chapter 3, “Basic Configuration for Open Server.” In order for the server and its clients to use DCE security services, you must perform these additional configuration steps:

1. Decide which DCE principal the server will run as.

You can run the server as the DCE root user, `./:/hosts/hostname/self`, where *hostname* is the computer where the server runs. You can also create a new principal for the server.

If necessary, use the DCE `dcecp` tool's `user create` command to create a new principal. The command options must specify that the new principal can act as a server.

2. If you do not run the server as the root principal, you must create a DCE keytab file for the server principal.

A DCE keytab file is an operating system file that contains a principal's password in an encrypted form. You can create a keytab file with the DCE `dcecp` utility, using the `keytab create` command. The keytab file must allow read permission for the operating system user who starts the Open Server. In a production environment, you must control the access to this file. If a user can read the keytab file, they can create a server that impersonates your server.

3. Make sure the DCE security driver is configured in the [SECURITY] section of `libtcl.cfg`.

See “SECURITY Section” on page B-4 for details.

4. When starting the server, specify the server principal name if it is not the same as the server's network name.

The Open Server's network name is its name in *interfaces* or DCE directory service. If the principal name does not match the network name, you must specify the principal name separately.

A custom Open Server application specifies the principal name by setting the `SRV_S_SEC_PRINCIPAL` Server-Library property.

Security Guardian users can specify the server's principal name with the `-R` command-line option.

5. When starting the server, specify the location of a DCE keytab file (see step 2 above) if the server is not run as the DCE root user (*./:/hosts/hostname/self*).

A custom Open Server application specifies the location of a keytab file by setting the `SRV_S_SEC_KEYTAB` Server-Library property.

Security Guardian users can specify the server's principal name with the `-K` command-line option.

Client-Library Applications and DCE Security

See "Client-Library and Security Services" on page 5-4 for an overview of how client applications use security services.

The following considerations apply specifically to client applications that use DCE security services:

- Client applications must specify the server principal name if it is not the same as the server's network name.

When using DCE security, DCE always authenticates the server's principal name. The connection cannot be opened if the correct server principal name is not supplied. By default, Client-Library assumes the principal name matches the network name.

Client-Library applications specify the server principal name by setting the `CS_SEC_SERVERPRINCIPAL` connection property. Users of `isql` and other Sybase client utilities can specify the server principal name with the `-R` command-line option.

- Client applications must connect to the server using the default, preexisting DCE credential or by using a DCE keytab file to acquire a new credential.

A DCE user acquires their default DCE credential with the DCE `dce_login` tool. Client-Library applications use the default credential by not setting the `CS_USERNAME` connection property. Users of Sybase client utilities, such as `isql`, omit the `-U` command-line option to connect using their default credential.

To acquire a new credential, you must have read access to a valid DCE keytab file that contains the encrypted password for the DCE user you want to connect as. You can create a keytab file with the DCE `dcecp` utility, using the `keytab create` command.

Client-Library applications can acquire a new credential by setting the `CS_USERNAME` property to a DCE user name and

setting the `CS_SEC_KEYTAB` property to the name of the corresponding DCE keytab file.

Users of `isql` and other Sybase client utilities can use the `-U` and `-K` command-line options to specify the user name and the keytab file name, respectively.

► **Note**

Do not specify a username unless you also supply the name of a DCE keytab file to authenticate that user.

E

CyberSAFE Kerberos Security Services

This appendix lists the security services supported by the CyberSAFE Kerberos security driver and summarizes some system configuration tasks that are required in order to use the CyberSAFE Kerberos security driver.

For an overview of the Open Client and Open Server security services architecture, see Chapter 5, “Using Security Services.”

Supported Security Services

The CyberSAFE Kerberos security mechanism provides the following services:

- Network authentication
- Mutual authentication
- Data integrity
- Data confidentiality
- Replay detection
- Out of sequence detection

See the *Open Client Client-Library/C Reference Manual* for a description of these security services.

Configuring CyberSAFE Kerberos for Use with Open Client and Open Server

This section summarizes some administration tasks that are required to use CyberSAFE Kerberos with your Open Client/Server applications.

Some administration tasks described here require you to use the CyberSAFE administration tools. See your CyberSAFE documentation for information on using these tools.

See *README.SEC* in the *SYBASE_home/sample/srvlibrary* directory for an example of configuring and running the example program.

Open Server Applications and CyberSAFE Kerberos

You can run a custom Open Server application or the Security Guardian server with CyberSAFE Kerberos security. In order for the server and its clients to communicate over the network, you must perform the normal configuration steps described in Chapter 3, “Basic Configuration for Open Server.” In order for the server and its clients to use CyberSAFE Kerberos security services, you must perform these additional configuration steps:

1. Decide which CyberSAFE Kerberos principal the server will run as.

You can create a new principal with the CyberSAFE `kadmin` utility, using the `add` command. The principal must be allowed to act as a server.

2. If the server principal does not already have a key in a CyberSAFE Kerberos server key table file, create one with the CyberSAFE `kadmin` utility, using the `ext` command. Make sure that the operating system user that starts the server has read permission on the server key table file. In a production environment, you must control the access to the key table file. If a user can read this file, they can create a server that impersonates your server.
3. Make sure the CyberSAFE Kerberos security driver is configured in the [SECURITY] section of `libtcl.cfg`. See “SECURITY Section” on page B-4 for details.
4. Set the `CSFC5KTNAME` environment variable to the name of the key table file that holds the key for the server principal (see item 2). The CyberSAFE runtime libraries require that this environment variable be set if the server key table file is in a location other than the CyberSAFE system default.
5. You must place the shared library file, `libgss.so` on Sun Solaris 2.x, `libgss.so.a` on IBM RS/6000, or `libgss.sl` on HP-UX, in a directory specified in the shared library path, `LD_LIBRARY_PATH` on Sun Solaris 2.x, `LIBPATH` on IBM RS/6000, and `SHLIB_PATH` on HP-UX.

This enables the client to find this shared library file at runtime. You can also place the shared library file in the `lib` subdirectory of the CyberSAFE installation as long as this subdirectory is in the shared library path.

This shared library is not provided by Sybase, but it is included in some CyberSAFE products. If it is not included with your

CyberSAFE product, contact CyberSAFE to obtain their GSS-API library.

6. When you start the server, specify the principal name in addition to the network name if the principal name does not match the network name. You do not have to specify the network name if you set the DSLISTEN environment variable to the network name.

The Open Server's network name is its name in *interfaces* or DCE directory service.

A custom Open Server application specifies the principal name by setting the SRV_S_SEC_PRINCIPAL Server-Library property.

Security Guardian users can specify the server's principal name with the -R command-line option.

When using CyberSAFE security, you cannot use the -K option to specify the key table; you must use the CSFC5KTNAME environment variable (see item 4).

Client-Library Applications and CyberSAFE Kerberos

See "Client-Library and Security Services" on page 5-4 for an overview of how client applications use security services. These considerations apply to client applications that use CyberSAFE Kerberos security services:

- The application must use a preexisting user credential to connect to the server. In other words, the user of the application must log into CyberSAFE before running the client application. On UNIX, use the CyberSAFE kinit utility to log into CyberSAFE.
- If a user name is supplied, it must match the user's preexisting credential. If a user name is not supplied, Client-Library connects to the server using the username associated with the user's CyberSAFE credential.

Index

A

Audience xiii
Auxiliary Open Server 3-1

B

bcp.loc file C-8
binary.srt file C-11
blklib.loc file C-8

C

charsets directory
 contents of C-3, C-8
Collating sequence files C-11
Connection
 Open Client 2-1
 Open Server 3-1
 overview of 1-2
Conversion configuration files
 entries in C-10 to C-11
 example of C-11
 how they are used C-9
 location of C-10
Creating DIT base entries
 dcecp B-4, B-8
cslib.loc file C-8
ctlib.loc file C-8
CyberSAFE Kerberos security
 configuration requirements E-1
 how to use in applications E-1

D

dcecp utility B-4, B-8
DCE directories
 editing with *dsedit_dce* 7-4
DCE security
 configuration requirements D-1
 using in your applications D-1

dictionary.srt file C-11
Directory drivers 4-3
 adding in *libtcl.cfg* file B-7
 ditbase B-3
 syntax in *libtcl.cfg* file B-3
Directory services
 See also dscp utility
 adding entries 6-8
 and the connection process 4-3 to 4-4
 attributes 4-2
 configuration tasks 4-5
 copying entries to 6-10 to 6-11
 deleting entries 6-9
 directory objects 4-2
 drivers for 4-3
 listing entries 6-7
 modifying entries 6-8
 opening a *dscp_dce* session to 6-4
 overview 4-1
 security attribute 5-2
 viewing an entry 6-7
Driver configuration file. *See libtcl.cfg* file
Drivers
 See also Directory drivers, Security
 drivers, Network drivers
 definition of B-2
 security 5-2
 types of B-1
dscp *dscp_dce* utilities
 about 6-1
 adding server entries 6-8
 and *libtcl.cfg* file 6-4
 commands 6-1 to 6-2
 copying server entries 6-10 to 6-11
 deleting server entries 6-9
 exiting 6-12
 help 6-3
 listing server entries 6-7
 modifying server entries 6-8
 starting 6-3
 viewing a server entry 6-7

dscp utility
 closing a session 6-5
 opening a session 6-4
 server attributes 6-6
 switching between sessions 6-4
dsedit and dsedit_dce utilities
 about 7-1

E

Environment variables
 for connection A-1
 for localization A-1
 setting A-2
esql.loc file C-8

G

Gateway Open Server 3-1

H

Help
 related documents xv

I

Initialization
 Open Client 2-1
 Open Server 3-1
 overview of process 1-2
Interfaces file
 editing with *dsedit* or *dsedit_dce* 7-2
interfaces file
See also dscp dscp_dce utility
 adding entries 6-8
 copying entries to 6-10 to 6-11
 deleting entries 6-9
 entries in, standard B-11 to B-14
 entries in, tli format B-14
 how it is used B-11
 listing entries 6-7
 location of B-11

modifying entries 6-8
 opening a *dscp* session with 6-4
 secmech line 5-2
 standby server addressing B-15
 tli format B-14
 viewing an entry 6-7

K

Kerberos. *See* CyberSAFE Kerberos security

L

libtcl.cfg file
 directory drivers in B-3
 how it is used B-2
 layout of B-3
 location of B-3
 network drivers in B-6
 sections B-3
 security drivers in B-4
locales.dat file
 editing C-6 to C-7
 entries in C-5
 file fragment C-6
 how it is used C-5
 location of C-5
locales directory
 contents of C-5, C-12
Localization
 overview of C-1 to C-2
Localization files
 about C-3
 collating sequence files C-11
 conversion configuration files C-9 to C-11
locales.dat file C-5 to C-7
 localized message files C-8
mnemonic.dat file C-13 to C-15
objectid.dat file C-12 to C-13
 Localized message files C-8

M

mnemonic.dat file
editing C-15
entries in C-14
example of C-14
how it is used C-13
location of C-14

N

Net-Library drivers. *See* Network drivers
Network drivers
adding in *libtcl.cfg* file B-9
syntax in *libtcl.cfg* file B-6
noaccents.srt file C-11
nocase.srt file C-11
nocasepref.srt file C-11

O

objectid.dat file
editing C-13
entries in C-12
file fragment C-12
location of C-12
ocs.cfg file B-16
Open Client
about 1-1
basic configuration for 2-1 to 2-4
configuration tasks 2-3
connection process 2-1
directory services 4-3
initialization process 2-1
localization process C-1 to C-2
security services 5-4
Open Server
about 1-1
basic configuration for 3-1 to 3-3
configuration tasks 3-3
connection process 3-1
directory services 4-3
initialization process 3-1

localization process C-1 to C-2
security services 5-5
types of applications 3-1, 4-4
oslib.loc file C-8

R

related documents xv

S

Security drivers 5-2
adding in *libtcl.cfg* file B-8
CyberSAFE Kerberos E-1
DCE D-1
syntax in *libtcl.cfg* file B-4
Security services
See also Security drivers and Client-Library 5-4
and Open Server 5-5
configuration tasks 5-6
example 5-3 to 5-4
overview of 5-1
provided by CyberSAFE Kerberos E-1
provided by DCE D-1
secmech line or attribute 5-2
security mechanisms 5-1 to 5-2
types of 5-2
Sort order files. *See* Collating sequence files
Syntax conventions xvii

