Programmers Supplement

Open Client™ and Open Server™

15.7

[UNIX]

DOCUMENT ID: DC35456-01-1570-02

LAST REVISED: June 2012

Contents

About This Book

The Sybase® Open Client™ and Open Server™ products are a set of programming interfaces that allow applications and data of any type to be used together. They include:

- Open Client DB-Library™/C
- Open Client Client-Library/C
- Open Server Server-Library/C
- Open Client Embedded SQL™/C
- Open Client Embedded SQL/COBOL

Each of these products has its own reference manual that describes it in detail. The purpose of this book is to serve as a supplement to the product manuals. It describes the platform-related issues for all the Open Client and Open Server products.

Audience

This book is written for programmers who use the Open Client and Open Server products listed above.

How to use this book

This book contains these chapters:

- Chapter 1, "Open Client Client-Library/C," provides information for building applications using Open Client and Open Server libraries.
- Chapter 2, "Open Client DB-Library/C," provides information on DB-Library sample programs and building an executable.
- Chapter 3, "Open Server Server-Library/C," provides information on Server-Library sample programs and building an executable.
- Chapter 4, "Open Client Embedded SQL/C," provides information on Embedded SQL/C sample programs and building an executable.
- Chapter 5, "Open Client Embedded SQL/COBOL,"provides information on Embedded SQL/Cobol sample programs and building an executable.
- Appendix A, "Utility Commands Reference," contains references pages that detail the syntax, parameters, and qualifiers for the commands and utilities relevant to Open Client.

- Appendix B, “Environment Variables,” provides information about the environment variables that have to be set to run build and applications.
- Appendix C, “Utility Messages,” provides information about error, information, and warning messages for the bcp, defncopy, and isql utilities.

Related documents

You can see these books for more information:

- The *Open Server and SDK New Features for Windows, Linux, and UNIX*, which describes new features available for Open Server and the Software Developer’s Kit. This document is revised to include new features as they become available.
- The *Open Server Release Bulletin* for your platform contains important last-minute information about Open Server.
- The *Software Developer’s Kit Release Bulletin* for your platform contains important last-minute information about Open Client™ and SDK.
- The *jConnect™ for JDBC™ Release Bulletin* contains important last-minute information about jConnect.
- The *Open Client and Open Server Configuration Guide* for your platform contains information about configuring your system to run Open Client and Open Server.
- The *Open Client Client-Library/C Programmers Guide* contains information on how to design and implement Client-Library applications.
- The *Open Client Client-Library/C Reference Manual* contains reference information for Open Client Client-Library™.
- The *Open Server Server-Library/C 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, which is a collection of utility routines that are useful in both Client-Library and Server-Library applications.
- The *Open Server DB-Library/C Reference Manual* contains reference information for the C version of Open Client DB-Library™.
- The *Installation and Release Bulletin Sybase® SDK DB-Library Kerberos Authentication Option* contains information about installing and enabling the MIT Kerberos security mechanism to be used on DB-Library. DB-Library only supports network authentication and mutual authentication in the Kerberos security mechanism.

- The *Open Client and Open Server International Developers Guide* provides information about creating internationalized and localized applications.
- The *Open Client Embedded SQL™/C Programmers Guide* explains how to use Embedded SQL and the Embedded SQL precompiler with C applications.
- The *Open Client Embedded SQL™/COBOL Programmers Guide* explains how to use Embedded SQL and the Embedded SQL precompiler with COBOL applications.
- The *jConnect for JDBC Programmers Reference* describes the jConnect for JDBC product and explains how to access data stored in relational database management systems.
- The *Adaptive Server® Enterprise ODBC Driver by Sybase® Users Guide* for Microsoft Windows and UNIX, provides information on how to access data from Adaptive Server on Microsoft Windows and UNIX platforms, using the Open Database Connectivity (ODBC) Driver.
- The *Adaptive Server Enterprise Database Driver for Perl Programmers Guide* provides information for Perl developers to connect to an Adaptive Server database and query or change information using a Perl script.
- The *Adaptive Server Enterprise extension module for PHP Programmers Guide* provides information for PHP developers to execute queries against an Adaptive Server database.
- The *Adaptive Server Enterprise extension module for Python Programmers Guide* provides information about Sybase-specific Python interface that can be used to execute queries against an Adaptive Server database.

Other sources of information

Use the Sybase Getting Started CD and the Sybase Product Documentation Web site to learn more about your product:

- The Getting Started CD contains release bulletins and installation guides in PDF format. It is included with your software. To read or print documents on the Getting Started CD, you need Adobe Acrobat Reader, which you can download at no charge from the Adobe Web site using a link provided on the CD.
- The Sybase Product Documentation Web site is accessible using a standard Web browser. In addition to product documentation, you will find links to EBFs/Maintenance, Technical Documents, Case Management, Solved Cases, newsgroups, and the Sybase Developer Network.

To access the Sybase Product Documentation Web site, go to Product Documentation at http://www.sybase.com/support/manuals/.

Sybase certifications on the Web

Technical documentation at the Sybase Web site is updated frequently.

❖ **Finding the latest information on product certifications**

1. Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.
2. Click Partner Certification Report.
3. In the Partner Certification Report filter select a product, platform, and timeframe and then click Go.
4. Click a Partner Certification Report title to display the report.

❖ **Finding the latest information on component certifications**

1. Point your Web browser to Availability and Certification Reports at http://certification.sybase.com/.
2. Either select the product family and product under Search by Base Product; or select the platform and product under Search by Platform.
3. Select Search to display the availability and certification report for the selection.

❖ **Creating a personalized view of the Sybase Web site (including support pages)**

Set up a MySybase profile. MySybase is a free service that allows you to create a personalized view of Sybase Web pages.

1. Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.
2. Click MySybase and create a MySybase profile.

Sybase EBFs and software maintenance

❖ **Finding the latest information on EBFs and software maintenance**

1. Point your Web browser to the Sybase Support Page at http://www.sybase.com/support.
2. Select EBFs/Maintenance. If prompted, enter your MySybase user name and password.
3. Select a product.

4 Specify a time frame and click Go. A list of EBF/Maintenance releases is displayed.

   Padlock icons indicate that you do not have download authorization for certain EBF/Maintenance releases because you are not registered as a Technical Support Contact. If you have not registered, but have valid information provided by your Sybase representative or through your support contract, click Edit Roles to add the “Technical Support Contact” role to your MySybase profile.

5 Click the Info icon to display the EBF/Maintenance report, or click the product description to download the software.

Conventions

Table 1 describes the syntax conventions used in this manual:

Table 1: Syntax conventions

Key	Definition
command	Command names, command option names, utility names, utility flags, and other keywords are in sans serif font.
variable	Variables, or words that stand for values that you fill in, are in *italics*.
{ }	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 items is optional. Do not include brackets in your option.
( )	Parentheses are to be typed as part of the command.
|	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.

Accessibility features

This document is available in an HTML version that is specialized for accessibility. You can navigate the HTML with an adaptive technology such as a screen reader, or view it with a screen enlarger.

Open Client and Open Server documentation has been tested for compliance with U.S. government Section 508 Accessibility requirements. Documents that comply with Section 508 generally also meet non-U.S. accessibility guidelines, such as the World Wide Web Consortium (W3C) guidelines for Web sites.

Note You might need to configure your accessibility tool for optimal use. Some screen readers pronounce text based on its case; for example, they pronounce ALL UPPERCASE TEXT as initials, and MixedCase Text as words. You might find it helpful to configure your tool to announce syntax conventions. Consult the documentation for your tool.

For information about how Sybase supports accessibility, see Sybase Accessibility at http://www.sybase.com/accessibility. The Sybase Accessibility site includes links to information on Section 508 and W3C standards.

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 documentation or online help, please have the designated person contact Sybase Technical Support or the Sybase subsidiary in your area.

CHAPTER 1 Open Client Client-Library/C

Open Client Client-Library is a collection of routines you can use to write client applications. Client-Library includes routines that send commands to a server and other routines that process the results of those commands. Other routines set application properties, handle error conditions, and provide a variety of information about an application's interaction with a server.

CS-Library, which is included with Open Client, is a collection of utility routines that you can use to write an Open Client or an Open Server application. All Client-Library applications include at least one call to CS-Library, because Client-Library routines use a structure that is allocated in CS-Library.

See the *Software Developer's Kit Release Bulletin* for the current release for additional information about Open Client products and how they behave on your platform.

See the *Open Server and SDK New Features for Windows, Linux, and UNIX* for a list of operating system platforms where the Open Client Client-Library/C is available.

General instructions

To run the Client-Library sample programs, you must:

- Be able to connect to an Adaptive Server® Enterprise. See the *Open Client and Open Server Configuration Guide for UNIX*. Also, see the descriptions of the individual samples for the required Adaptive Server version level.

- Set these environment variables, which are described in Appendix B, “Environment Variables”:
    - SYBASE
    - SYBASE_OCS
    - DSQUERY
    - SYBPLATFORM
    - Platform-specific library path variable
- Read the *README* file in *$SYBASE/$SYBASE_OCS/sample/ctlibrary* directory for complete instructions on running the sample programs.

Building a Client-Library executable

Use the libraries and compile-and-link lines to build Client-Library applications, including multithreaded applications.

Table 1-1 lists the libraries that you need to include to take full advantage of all Client-Library capabilities in a nonthreaded environment.

Table 1-1: Libraries for non-threaded environment

Platform	Required libraries
All platforms	*libsybct* – Client-Library (Sybase)
	libsybcs – CS-Library (Sybase)
	libsybtcl – transport control layer (Sybase internal)
	libsybcomn – an internal shared utility library (Sybase internal)
	libsybintl – internationalization support library (Sybase internal)
	libsybunic – Unicode-Library (Sybase internal)

Native thread support

The Client-Library version includes thread-safe libraries that allow developers to create multithreaded applications using POSIX threads.

See “Compile-and-link lines for multithreaded applications” on page 7 for proper syntax and examples.

Table 1-2 lists the libraries that you need to include to take advantage of all Client-Library capabilities for multithreaded support.

Table 1-2: Platform-specific libraries for multithreaded support

Platform	Required libraries
All platforms	*libsybct_r* – Client-Library (Sybase) *libsybcs_r* – CS-Library (Sybase) *libsybintl_r* – internationalization support library (Sybase internal) *libsybtcl_r* – transport control layer (Sybase internal) *libsybcomn_r* – internal shared utility library (Sybase internal)
Solaris platforms	*libthread* – native thread library (system) *libpthread* – thread library (system) *libsocket* – socket network library (system) *libnsl* – a network library (system) *libdl* – dynamic loader library (system)
HP HP-UX platforms	*libcl* – HP transport control layer (system) *libBSD* – the BSD library (system) *libc_r* – C reentrant library *libdld* – dynamic loader library (system)
IBM AIX platforms	*libc_r* – C reentrant library *libpthreads* – thread library (system)
Linux platforms	*libpthread* – thread library (system)

Kerberos support

Client-Library supports Kerberos security features for applications that require a high level of security when communicating over a network. By installing the required Kerberos software, and performing the appropriate configuration tasks, your Client-Library applications can take advantage of the following Kerberos security features:

- Network authentication
- Mutual authentication
- Out-of-sequence authentication
- Replay detection
- Confidentiality
- Integrity
- Credential delegation

Table 1-3: Required tasks for Kerberos support

Tasks	For more information
Install the Kerberos software on your system.	See your Kerberos documentation and the *Open Client and Open Server Configuration Guide for UNIX*.
Configure the security section of the *libtcl.cfg* configuration file.	See the *Open Client and Open Server Configuration Guide for UNIX*.
Log in to the Kerberos security environment with the Kerberos kinit utility, before running your Client-Library application.	See your Kerberos documentation.
Set the environment variable to the credential cache directory location.: • For CyberSafe, CSFC5CCNAME • For MIT, KRB5CCNAME	See your Kerberos documentation. Default credential cache directory location varies by platform.
Set the desired security features using ct_con_props or use the default credentials by not setting ct_con_props	See the *Open Client Client-Library/C Reference Manual*. Use CS_SUPPORTED action type in ct_con_props and ct_config to determine if a security feature is supported.

Compile-and-link lines

Client-Library and Server-Library dynamically link directory drivers and security drivers. Do not explicitly link the Sybase directory or security drivers (linker options -lsybdldap and -lsybskrb) with your applications.

Compile-and-link lines for non-threaded applications

The following tables list the general forms of the commands for compiling and linking non-threaded Client-Library applications on Sybase-supported platforms running on UNIX. Also, see the *makefile* and *sybopts.sh* file in *$SYBASE/$SYBASE_OCS/sample/ctlibrary* for compile and link information.

Table 1-4 shows commands for compiling and linking Client-Library applications using static libraries.

Table 1-4: Static compile-and-link commands for Client-Library

Platform	Command
Solaris SPARC 32-bit and 64-bit	`/opt/SUNWspro6.2/bin/cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Bstatic -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -Bdynamic -lnsl -ldl -lm -lsocket -o program`
Solaris x86-64 32-bit and 64-bit	`/opt/SunStudio10/SUNWspro/bin/cc -xtarget=opteron -xarch=amd64 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Bstatic -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -Bdynamic -lnsl -ldl -lm -lsocket -o program`
IBM AIX RS/6000 32-bit and 64-bit	`xlc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bstatic -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lm -o program`
HP HP-UX PA-RISC 32-bit and 64-bit	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,a,archive -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -Wl,-a,default -lcl -lm -lBSD -ldld -Wl,-E,+s -o program`
HP HP-UX Itanium 32-bit and 64-bit	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,a,archive -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -Wl,-a,default -lcl -lm -lBSD -ldld -Wl,-E,+s -o program`
Linux x86 32-bit	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bstatic -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -Wl,-Bdynamic -ldl -lnsl -lm -o program`
Linux POWER 32-bit and 64-bit	`xlc -q32 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bstatic -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -Wl,-Bdynamic -ldl -lnsl -lm -o program`
Linux x86-64 64 bit	`gcc -I$SYBASE/$SYBASE_OCS/include L$SYBASE/$SYBASE_OCS/lib program.c -lsybct64 -lsybcs64 -lsybtcl64 -lsybcomn64 -lsybintl64 -lsybunic64 -lld -lnsl -lm64 -o program`

Table 1-5 shows commands for compiling and linking Client-Library applications using debug libraries.

Table 1-5: Debug compile-and-link commands for Client-Library

Platform	Command
Solaris SPARC 32-bit and 64-bit	`/opt/SUNWspro/bin/cc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lnsl -ldl -lm -lsocket -o program`
Solaris x86-64 32-bit and 64-bit	`/opt/SunStudio10/SUNWspro/bin/cc -xtarget=opteron -xarch=amd64 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lnsl -ldl -lm -lsocket -o program`
IBM AIX RS/6000 32-bit and 64-bit	`xlc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lm -o program`
HP HP-UX PA-RISC 32-bit and 64-bit	`cc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lcl -lm -lBSD -ldld -o program`
HP HP-UX Itanium 32-bit and 64-bit	`cc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lcl -lm -lBSD -ldld -o program`
Linux x86 32-bit	`cc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -ldl -lnsl -lm -o program`
Linux POWER 32-bit and 64-bit	`xlc -q32 -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -ldl -lnsl -lm -o program`
Linux x86-64 64 bit	`gcc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -lsybct64 -l sybcs64 -lsybtcl64 -lsybcomn64 -lsybintl64 -lsybunic64 -lld -lnsl -lm64 -o program`

Table 1-6 shows commands for compiling and linking Client-Library applications using shareable libraries (with dynamic drivers).

Table 1-6: Shareable compile-and-link commands for Client-Library

Platform	Command
Solaris SPARC 32-bit and 64-bit	`/opt/SUNWspro/bin/cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -R$SYBASE/$SYBASE_OCS/lib program.c -Bdynamic -lsybct -lsybcs -lnsl -ldl -lm -lsocket -o program`
Solaris x86-64 32-bit and 64-bit	`/opt/SunStudio10/SUNWspro/bin/cc -xtarget=opteron -xarch=amd64 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -R$SYBASE/$SYBASE_OCS/lib program.c -Bdynamic -lsybct -lsybcs -lnsl -ldl -lm -lsocket -o program`
IBM AIX RS/6000 32-bit and 64-bit	`xlc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bdynamic -lsybct -lsybcs -lm -o program`
HP HP-UX PA-RISC 32-bit and 64-bit	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,a,shared_archive -lsybct -lsybcs -lcl -lm -lBSD -o program`
HP HP-UX Itanium 32-bit and 64-bit	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,a,shared_archive -lsybct -lsybcs -lcl -lm -lBSD -o program`
Linux x86 32-bit	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bdynamic -lsybct -lsybcs -ldl -lnsl -lm -o program`
Linux POWER 32-bit and 64-bit	`xlc -q32 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bdynamic -lsybct -lsybcs -ldl -lnsl -lm -o program`
Linux x86-64 64-bit	`gcc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bdynamic -lsybct64 -lsybcs64 -ldl -lnsl -lm64 -o program`

Compile-and-link lines for multithreaded applications

Table 1-7 shows commands for compiling and linking Client-Library applications with libraries to take advantage of thread-safe support.

Table 1-7: Thread-safe compile-and-link commands for Client-Library

Platform	Command
Solaris SPARC 32-bit and 64-bit	`/opt/SUNWspro/bin/cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -D_REENTRANT program.c -lsybct_r -lsybcs_r -lsybtcl_r -lsybcomn_r -lsybintl_r -lnsl -ldl -lpthread -lthread -lm -lsocket -o program`
Solaris x86-64 32-bit and 64-bit	`/opt/SunStudio10/SUNWspro/bin/cc -xtarget=opteron -xarch=amd64 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -D_REENTRANT program.c -lsybct_r -lsybcs_r -lsybtcl_r -lsybcomn_r -lsybintl_r -lnsl -ldl -lpthread -lthread -lm -lsocket -o program`
IBM AIX RS/6000 32-bit and 64-bit	`xlc_r -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -D_THREAD_SAFE program.c -lsybct_r -lsybcs_r -lsybtcl_r -lsybcomn_r -lsybintl_r -lpthread -lm -o program`
HP HP-UX PA-RISC 32-bit and 64-bit	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -D_THREAD_SAFE -D_REENTRANT -Ae program.c -lsybct_r -lsybcs_r -lsybtcl_r -lsybcomn_r -lsybintl_r -lcl -lm -lBSD -lpthread -ldld -o program`
HP HP-UX Itanium 32-bit and 64-bit	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -D_THREAD_SAFE -D_REENTRANT -Ae program.c -lsybct_r -lsybcs_r -lsybtcl_r -lsybcomn_r -lsybintl_r -lcl -lm -lBSD -lpthread -ldld -o program`
Linux x86 32-bit	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -lsybct_r -lsybcs_r -lsybtcl_r -lsybcomn_r -lsybintl_r -ldl -lpthread -lnsl -lm -o program`
Linux POWER 32-bit and 64-bit	`xlc_r -q32 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -lsybct_r -lsybcs_r -lsybtcl_r -lsybcomn_r -lsybintl_r -ldl -lpthread -lnsl -lm -o program`

Platform	Command
Linux x86-64 64-bit	`gcc_r -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -lsybct64_r -lsybcs64_r -lsybtcl64_r -lsybcomn64_r -lsybintl64_r -ldl -lpthread -lnsl -lm64 -o program`

For HP HP-UX system users:

- The option -Wl,-a,archive causes the linker to statically link the Sybase libraries. If you do not specify this option, Client-Library uses shared versions of the Sybase libraries. When using shared libraries, the SHLIB_PATH environment variable must include *$SYBASE/$SYBASE_OCS/lib* at runtime, and the application user must have read and execute permission on the libraries in *$SYBASE/$SYBASE_OCS/lib*.
- HP HP-UX does not use the SHLIB_PATH environment variable at runtime unless the application is linked with the +s linker option. You must use the +s linker option so that the system can find Sybase libraries at runtime. -E is required to prevent undefined-symbol errors when driver libraries are loaded at runtime. See the HP-UX ld man page.
- Set the environment variable LD_LIBRARY_PATH to *$SYBASE/$SYBASE_OCS/lib* to run programs linked with shareable (dynamic) libraries. If you are running in debug mode, set LD_LIBRARY_PATH to *$SYBASE/$SYBASE_OCS/devlib* to run the program.

  LD_LIBRARY_PATH is platform-specific; Table 1-8 lists the environment variables for each platform.

Table 1-8: LD_LIBRARY_PATH for UNIX platforms

Platform	Environment variable
Solaris SPARC 32-bit, Solaris x86-64 32-bit, HP HP-UX PA-RISC 64-bit, HP HP-UX Itanium 64-bit, Linux x86 32-bit, Linux POWER 32-bit and 64-bit, Linux x86-64 64-bit	LD_LIBRARY_PATH
Solaris SPARC 64-bit, Solaris x86-64 64-bit	LD_LIBRARY_PATH_64
HP HP-UX PA-RISC 32-bit, HP HP-UX Itanium 32-bit	SHLIB_PATH
IBM AIX RS/6000 32-bit and 64-bit	LIBPATH

Compile-and-link lines for Kerberos-supported applications

The Sybase driver for Kerberos is a dynamically loaded shared library. When the driver is loaded, it attempts to dynamically load a Kerberos GSS library, which must be in the search path that the dynamic loader uses. Due to constraints in the implementation of the Sybase driver, only reentrant libraries are supported when using Kerberos.

Bulk-copy routines

To use bulk copy routines, link in the *libsybblk* library. To use bulk-copy routines in a threaded applications, link in the *libsybblk_r* library.

To link in the bulk-copy library:

- In nonthreaded applications, add -lsybblk before -lsybct on the link line.
- In multithreaded applications, add -lsybblk_r before -lsybct_r on the link line.

See the *Open Client and Open Server Common Libraries Reference Manual*.

Performance considerations

Linking with shared libraries results in a smaller executable and takes less time than linking with static libraries. However, executables that link with shared libraries may have a slower start-up time than those that link with static libraries. Unlike static libraries, shared libraries must be available at runtime.

The type of library that provides the best performance is determined by your individual site requirements.

Header files

Include the *ctpublic.h* header file in all Client-Library application source files. Other necessary header files are nested in *ctpublic.h*. If Bulk-Library is used, include *bkpublic.h* instead of *ctpublic.h*.

See the *Open Client Client-Library/C Reference Manual*.

Using Client-Library sample programs

Sample programs are included with Client-Library to demonstrate typical uses for Client-Library routines.

Some sample programs use the sample databases supplied with Adaptive Server. See the *Installation Guide Adaptive Server Enterprise 15.7* of your platform for information on installing the sample databases. The requirements section for each sample lists the database you need, if any.

makefile and sample programs

To use the *makefile* to build sample programs on all platforms, you must correctly set the SYBPLATFORM environment variable for the compiler you are using. See Table B-1 on page 158.

Purpose of the sample programs

The sample programs demonstrate specific Client-Library functionality. These programs are designed as guides for application programmers, not as Client-Library training aids. Read the descriptions at the top of each source file, and examine the source code prior to using the sample programs.

Note These simplified programs are not intended for use in a production environment. Production-quality programs require additional code to handle errors and special cases.

The sybopts.sh script and building applications

The *sybopts.sh* reads the SYBPLATFORM environment variable to help you build Open Client and Open Server applications:

```
sybopts.sh args
```

where *args* can be, for example:

- compile – returns the compiler command and platform-specific compile flags.

- comlibs – returns the list of required Sybase libraries that must be linked with the application.
- syslibs – returns the list of required non-Sybase system libraries that must be linked with the application.

For a complete list of arguments (*args*), see the “Usage” section in the *sybopts.sh* script in *$SYBASE/$SYBASE_OCS/sample/ctlibrary*.

Location

The sample programs are located in *$SYBASE/$SYBASE_OCS/sample/ctlibrary*.

This directory includes:

- Source code for the sample programs.
- Data files for the samples.
- The *makefile* provided to build the samples. Use the *makefile* as a starting point for your own Client-Library applications.
- The samples header files — *example.h*, *ctxact.h*, *exasync.h*, *exutils.h*, *thrdfuc.h*, *thrdutil.h*, and *wide_example.h*
- The *README* file containing instructions for building, executing, and testing the samples.

Before compiling and running the sample programs, copy the contents of *$SYBASE/$SYBASE_OCS/sample/ctlibrary* into a working directory, where you can experiment with the sample programs without affecting the integrity of the original files.

Header file

All of the sample programs reference the sample header file, *example.h*, the contents of which are as follows:

```
/*
 ** example.h
 **
 ** This is the header file that goes with the
 ** Sybase Client-Library sample programs.
 **
```

```
**
*/

/*
** Define symbolic names, constants, and macros
*/
#define EX_MAXSTRINGLEN     255
#define EX_BUFSIZE          1024
#define EX_CTLIB_VERSION    CS_CURRENT_VERSION
#define EX_BLK_VERSION      BLK_VERSION_155
#define EX_ERROR_OUT        stderr

/*
** exit status values
*/
#define  EX_EXIT_SUCCEED 0
#define  EX_EXIT_FAIL 1

/*
** Define global variables used in all sample
** programs
*/
#define EX_SERVER           NULL/* use DSQUERY
                                  env var */
#define EX_USERNAME         "sa"
#define EX_PASSWORD         ""
```

The sample programs make use of the define statements in *example.h* as illustrated in the following fragments:

```
CS_CHAR *Ex_username = EX_USERNAME;
CS_CHAR *Ex_password = EX_PASSWORD;

/*
** If a user name is defined, set the
** CS_USERNAME property.
*/
if (retcode == CS_SUCCEED && Ex_username != NULL)
{
    if ((retcode = ct_con_props(*connection,
        CS_SET, CS_USERNAME, Ex_username,
        CS_NULLTERM, NULL)) != CS_SUCCEED)
    {
        ex_error("ct_con_props(username) failed");
    }
}
```

```
/*
** If a password is defined, set the
** CS_PASSWORD property.
*/
if (retcode == CS_SUCCEED && Ex_password != NULL)
{
    if ((retcode = ct_con_props(*connection,
        CS_SET, CS_PASSWORD, Ex_password,
        CS_NULLTERM, NULL)) != CS_SUCCEED)
    {
        ex_error("ct_con_props(password) failed");
    }
}
```

EX_USERNAME is defined in *example.h* as “sa.” Before running the sample programs, edit *example.h* to change “sa” to your server login name.

EX_PASSWORD is defined in *example.h* as a null (“ ”) string. Before running the sample programs, you may want to edit *example.h* and change the null (“ ”) string to your server password.

There are three options regarding EX_PASSWORD. Choose the one that best meets your needs:

- Change your server password to a null (“ ”) string while you are running the samples. This creates the possibility of a security breach, because while your password is set to this published value, an unauthorized person might take the opportunity to log in to the server as you. If this is a problem, choose one of the other methods of handling passwords for the sample programs.
- In *example.h*, change the null (“ ”) string to your own server password. Use the operating system’s protection mechanisms to prevent others from accessing the header file while you are using it. When you are finished with the samples, edit the line so that it again says “server_password.”
- In the sample programs, modify the ct_con_props code that sets the server password—substitute your own code to prompt users of the samples for their server passwords. Because this code is platform-specific, Sybase does not supply it.

Utility routines for the sample programs

The *exutils.c* file contains utility routines that are used by all other Client-Library sample programs. It demonstrates how an application can hide some implementation details of Client-Library from a higher-level program.

For more information about these routines, see the leading comments in the sample source file.

The *wide_util.c* file contains these generic routines that are used by the wide_* sample programs:

- The init_db routine allocates the context and initializes the library. It also installs the callback routines and is called at the beginning of several sample programs.
- The cleanup_db routine closes the connection to the server and cleans up the context structure. This function is called at the end of the *wide_curupd.c* and *wide_dynamic.c* sample programs.
- The connect_db routine connects to the server, then sets the appropriate user name and password.
- The handle_returns routine processes the return result type.
- The fetch_n_print routine fetches the bound data into a host variable.

Sample program summaries

Unless otherwise specified, see the leading comments in the source files for additional information about each of the sample programs.

arraybind.c sample program

The *arraybind.c* sample program demonstrates how to use array binding with a CS_LANG_CMD initiated by ct_command. The sample program uses a hard-coded query of a hard-coded table in the pubs2 database. This query is defined by a language command using a select statement. The *arraybind.c* program then processes the results using the standard ct_results while loop. It binds column values to program arrays, then fetches and displays the rows in the standard ct_fetch loop.

Note This sample requires the pubs2 database.

batch_lang.c sample program

The *batch_lang.c* sample program demonstrates how ct_send_params() can be used with a language statement. This sample uses ct_send_params() repeatedly to insert lines read from a file into a table. Since it uses the same location for the parameters for every line read, it does not need to call ct_param() or ct_setparam() in between calls to ct_send_params().

batch_dynamic.c sample program

The *batch_dynamic.c* sample program uses dynamic SQL and sends parameters to the server for which the data resides at different memory locations. Therefore, this sample also demonstrates how ct_setparam() can be used to rebind to different variables before calling ct_send_params() again.

blktxt.c sample program

The *blktxt.c* sample program uses the bulk-copy routines to copy static data to a server table. There are three rows of data that are bound to program variables and then sent to the server as a batch. The rows are again sent using blk_textxfer to send the text data.

compute.c sample program

The *compute.c* sample program demonstrates how computed results are processed:

- Sends a query to the server using a language command.
- Processes the results using the standard ct_results while loop.
- Binds the column values to program variables.
- Fetches and displays the rows in the standard ct_fetch while loop.

Note This sample requires the pubs2 database.

This is the query that is sent to the server:

```
select type, price from titles
where type like "%cook"
order by type, price
compute sum(price) by type
compute sum(price)
```

This query returns both regular rows and computed rows. The computed rows are generated by the two compute clauses.

- The first compute clause generates a compute row each time the value of type changes:

  ```
  compute sum(price) by type
  ```

- The second compute clause generates one compute row, which is the last to be returned:

  ```
  compute sum(price)
  ```

csr_disp.c sample program

The *csr_disp.c* sample program demonstrates how to use a read-only cursor:

- It opens a cursor with a query.
- It processes the results using the standard ct_results while loop.
- It binds the column values to program variables.
- It fetches and displays the rows in the standard ct_fetch while loop.

Note This sample requires the pubs2 database.

This is the query:

```
select au_fname, au_lname, postalcode
from authors
```

csr_disp_scrollcurs.c sample program

The *csr_disp_scrollcurs.c* sample program uses a scrollable cursor to retrieve data from the authors table in the pubs2 database:

- Sends a query to the server to open a cursor.
- Processes the results using the standard ct_results while loop.
- Binds the column values to program variables.
- Fetches and displays the rows in the standard ct_scroll_fetch while loop.

Note This example requires Adaptive Server version 15.0 or later, with scrollable cursor support, and the pubs2 database.

This example uses a single prefetch buffer and regular program variables. This is the query:

```
select au_fname, au_lname, postalcode
from authors
```

csr_disp_scrollcurs2.c sample program

The *csr_disp_scrollcurs2.c* sample program uses a scrollable cursor to retrieve data from the authors table in the pubs2 database:

- Sends a query to the server to open a cursor.
- Processes the results using the standard ct_results while loop.
- Binds the column values to program variables.
- Fetches the rows using ct_scroll_fetch and displays them.

Note This example requires Adaptive Server version 15.0 or later, with scrollable cursor support, and the pubs2 database.

This example uses a scrollable cursor with arrays as program variables and array binding. A single ct_scroll_fetch call displays results in an array.

This is the query:

```
select au_fname, au_lname, postalcode
from authors
```

csr_disp_implicit.c sample program

The *csr_disp_implicit.c* sample program demonstrates how to use an implicit read-only cursor. It:

- Opens a cursor with a query.
- Processes the results using the standard ct_results while loop.
- Binds the column values to program variables.
- Fetches and displays the rows in the standard ct_fetch while loop.

Note This example requires Adaptive Server version 12.5.1 or later and the pubs2 database.

The program flow is the same as the *csr_disp.c* sample program, with the only difference being the usage of the CS_IMPLICIT_CURSOR option instead of CS_READ_ONLY in the first ct_cursor call. Although, the generated output is the same as the *csr_disp.c* example, the use of CS_IMPLICIT_CURSOR potentially reduces network traffic at the network level.

When using this example, set the CS_CURSOR_ROWS option to a value greater than 1.

This is the query:

```
select au_fname, au_lname, postalcode
from authors
```

ex_alib.c and *ex_amain.c* sample programs

This sample program demonstrates how to write an asynchronous layer on top of Client-Library. It uses hooks provided by Client-Library to allow seamless polling and use of completion callbacks.

The sample program is composed of two files:

- *ex_alib.c* contains the source code to the library portion of the example. It is meant to be part of a library interface that supports asynchronous calls. ex_alib.*c* sends a query to, retrieves results from, a server using only one asynchronous operation.
- *ex_amain.c* contains the source code to the main program that uses the services provided by *ex_alib.c*.

See the leading comments in both the example source files and in the *EX_AREAD.ME* file.

exconfig.c sample program

The *exconfig.c* sample program demonstrates how to externally configure Client-Library application properties.

This sample requires you to edit the default runtime configuration file, *$SYBASE/$SYBASE_OCS/config/ocs.cfg*. The example sets the CS_CONFIG_BY_SERVERNAME Client-Library property and calls ct_connect with a *server_name* parameter set to “server1.” In response, Client-Library looks for a [server1] section in the external configuration file. To run the example, create *$SYBASE/$SYBASE_OCS/config/ocs.cfg* (if necessary) and add the section:

```
[server1]
```

```
CS_SERVERNAME = real_server_name
```

where *real_server_name* is the name of the server that you want to connect to.

For more information on how Client-Library uses external configuration files, see “Using the Runtime Configuration File” in the *Open Client Client-Library/C Reference Manual*.

firstapp.c sample program

The *firstapp.c* sample program is an introductory example that connects to the server, sends a select query, and prints the rows. This sample program is described in the *Open Client Client-Library/C Programmers Guide*.

getsend.c sample program

The *getsend.c* sample program demonstrates how to retrieve and update text data from a table containing various datatypes. You can use the same process as demonstrated to retrieve and update image data.

i18n.c sample program

The *i18n.c* sample program demonstrates some of the international features available in Client-Library, including:

- Localized error messages
- User-defined bind types

multthrd.c and *thrdfunc.c* sample programs

This sample program demonstrates a multithreaded Client-Library application. The program contains two files:

- *multthrd.c* contains source code that spawns five threads. Each thread processes a cursor or a regular query. The main thread waits for the other threads to complete query processing and then terminates.
- *thrdfunc.c* contains platform-specific information that determines which thread and synchronization routines the sample uses for execution.

This sample cannot run if your platform does not support a complete POSIX thread implementation. You must set the SYBPLATFORM environment variable described in Appendix B, “Environment Variables.”

rpc.c sample program

The RPC command sample program, *rpc.c*, sends an RPC command to a server and processes the results.

secct.c sample program

The *secct.c* sample program demonstrates how to use network-based security features in a Client-Library application.

For this sample to execute, Kerberos must be installed and running on your machine. You must also connect to a server that supports network-based security, such as Adaptive Server or the *secsrv.c* Open Server sample program.

For more information about network security services, see the *Open Client and Open Server Configuration Guide for UNIX*.

uni_blktxt.c sample program

The *uni_blktxt.c* sample program uses the bulk-copy routines, including unichar and univarchar datatypes, to copy static data to a server table. There are three rows of data that are bound to program variables and then sent to the server as a batch. The rows are sent a second time using blk_textxfer to send the text data.

uni_compute.c sample program

The *uni_compute.c* sample program demonstrates how to process compute results. It is a modification of the *compute.c* sample program for the unichar and univarchar datatypes and requires the unipubs2 database: It:

- Sends a query to the server using a language command.
- Processes the results using the standard ct_results loop.
- Binds the column values to program variables.
- Fetches the rows using ct_fetch loop and displays them.

For instructions on installing the unipubs2 database, see the *README* file in *$SYBASE/$SYBASE_OCS/sample/ctlibrary*.

uni_csr_disp.c **sample program**

The *uni_csr_disp.c* sample program demonstrates how to use a read-only cursor. It is a modification of the *csr_disp.c* sample program and requires the unipubs2 database. It:

- Opens a cursor with a query.
- Processes the results using the standard ct_results while loop.
- Binds the column values to program variables.
- Fetches and displays the rows in the standard ct_fetch while loop.

This is the query:

```
select au_fname, au_lname, postalcode
from authors
```

For instructions on installing the unipubs2 database, see the *README* file in *$SYBASE/$SYBASE_OCS/sample/ctlibrary*.

uni_firstapp.c **sample program**

This is a modification of the *firstapp.c* sample program for use with unichar and univarchar datatypes. It is an introductory example that connects to the server, sends a select query, and prints the rows. The *firstapp.c* program is described in the *Open Client Client-Library/C Programmers Guide*.

uni_rpc.c **sample program**

The RPC command sample program, *uni_rpc.c*, sends an RPC command to a server and processes the results. This is a modification of the *rpc.c* sample program for use with unichar and univarchar datatypes, and requires the unipubs2 database. For instructions on installing the unipubs2 database, read the *README* file in *$SYBASE/$SYBASE_OCS/sample/ctlibrary*.

usedir.c **sample program**

The *usedir.c* sample program demonstrates the ability of Client-Library to query a directory service for a list of available servers.

usedir.c searches for Sybase server entries in the default directory, as defined in the driver configuration file. If a network directory service is not used, *usedir.c* queries the interfaces file for server entries. Then, it displays a description of each entry found, and lets the user choose a server to connect to.

For more information about network directory services, see the *Open Client and Open Server Configuration Guide for UNIX*.

wide_compute.c sample program

The *wide_compute.c* sample program demonstrates how to process compute results with wide tables and larger column sizes. It:

- Sends a query to the server using a language command.
- Processes the results using the standard ct_results while loop.
- Binds the column values to program variables.
- Fetches and displays the rows in the standard ct_fetch while loop.

Note This sample requires the pubs2 database.

This is the query:

```
select type, price from titles
where type like "%cook"
order by type, price
compute sum(price) by type
compute sum(price)
```

This query returns both regular rows and rows that are returned by a compute clause. The computed rows are generated by the two compute clauses:

- The first compute clause generates a compute row each time the value of type changes:

  ```
  compute sum(price) by type
  ```

- The second compute clause generates one compute row, which is the last to be returned:

  ```
  compute sum(price)
  ```

wide_curupd.c sample program

The *wide_curupd.c* sample program uses a cursor to retrieve data from the table called “publishers” in the pubs2 database. It retrieves data row by row and prompts the user to input new values for the column state in the publishers table.

Inputs value for the input parameter (state column from the publishers table) for the UPDATE. Create a publishers3 table as shown before running the sample program:

```
use pubs2

go

drop table publishers3

go

create table publishers3 (pub_id char(4) not null,
    pub_name varchar(400) null, city varchar(20) null,
    state char(2) null)

go

select * into publishers3 from publishers

go

create unique index pubind on publishers3(pub_id)

go
```

wide_dynamic.c sample program

The *wide_dynamic.c* sample program uses a cursor to retrieve data from the table called "publishers" in the pubs2 database. It retrieves data row by row and prompts the user to input new values for the column called "state" in the publishers table.

This program uses dynamic SQL to retrieve values from the titles table in the tempdb database. The select statement, which contains placeholders with identifiers, is sent to the server to be partially compiled and stored. Therefore, every time you call the select statement, you pass only new values for the key value, which determines the row to be retrieved. The behavior is similar to passing input parameters to stored procedures. The program also uses cursors to retrieve rows one by one, which can be manipulated as required.

wide_rpc.c sample program

The RPC command sample program, *rpc.c*, sends an RPC command to a server and processes the results. This is the same as the *rpc.c* program, but it uses wide tables and larger column sizes.

CHAPTER 2 Open Client DB-Library/C

Open Client DB-Library is a collection of routines you can use to write client applications. DB-Library is the predecessor to Client-Library. Some functionality, such as Directory and Security services support, is not included with DB-Library. You must use Client-Library to take advantage of these services.

DB-Library includes routines that send commands to a server and others that process the results of those commands. Other routines set application properties, handle error conditions, and provide a variety of information about an application's interaction with a server.

See the *Open Server and SDK New Features for Windows, Linux, and UNIX* for a list of operating system platforms where the Open Client DB-Library/C is available.

General instructions

To run DB-Library applications, including the sample programs, you must:

- Set these environment variables, which are described in Appendix B, "Environment Variables":
    - SYBASE
    - SYBASE_OCS
    - DSQUERY
    - SYBPLATFORM
    - Platform-specific library path variable

- Be able to connect to an Adaptive Server database. See the *Open Client and Open Server Configuration Guide for UNIX* .
- Read the *README* file in each product directory under *$SYBASE/$SYBASE_OCS/sample/dblibrary*. Complete instructions for running the samples are in the *README* file.
- Set execute permission on the *sybopts.sh* file for the file's owner:

```
chmod u+x sybopts.sh
```

Building a DB-Library executable

Use libraries, linking, and header files to build a DB-Library executable.

Libraries

Include the libraries for all platforms to take full advantage of all DB-Library capabilities:

- *libsybdb* – DB-Library (Sybase)
- *libsybunic* – Unicode-Library (Sybase)

Compile-and-link lines

Table 2-1 and Table 2-2 list the general forms of the commands for compiling and linking DB-Library applications on Sybase-supported platforms running the UNIX operating system. Table 2-1 shows the commands for compiling and linking DB-Library applications using static libraries.

Table 2-1: Static compile-and-link commands for DB-Library

Platform	Command
Solaris SPARC 32-bit and 64-bit	`/opt/SUNWspro/bin/cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Bstatic -lsybdb -lsybunic -o program`
Solaris x86-64 32-bit and 64-bit	`/opt/SunStudio10/SUNWspro/bin/cc -xtarget=opteron -xarch=amd64 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Bstatic -lsybdb -lsybunic -o program`
IBM AIX RS/6000 32-bit and 64-bit	`xlc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bstatic -lsybdb -lsybunic -o program`
HP HP-UX PA-RISC 32-bit and 64 bit	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-a,archive -lsybdb -lsybunic -Wl,-E,+s -o program`
HP HP-UX Itanium 32-bit and 64-bit	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-a,archive -lsybdb -lsybunic -Wl,-E,+s -o program`
Linux x86 32-bit	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bstatic -lsybdb -lsybunic -ldl -o program`
Linux POWER 32-bit and 64-bit	`xlc -q32 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bstatic -lsybdb -lsybunic -ldl -o program`
Linux x86-64 64-bit	`gcc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bstatic -lsybdb64 -lsybunic64 -ldl -o program`

Table 2-2 shows the commands for compiling and linking DB-Library applications using debug libraries.

Table 2-2: Debug compile-and-link commands for DB-Library

Platform	Command
Solaris SPARC 32-bit and 64-bit	`/opt/SUNWspro/bin/cc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybdb -lsybunic -o program`
Solaris x86-64 32-bit and 64-bit	`/opt/SunStudio10/SUNWspro/bin/cc -xtarget=opteron -xarch=amd64 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybdb -lsybunic -o program`
IBM AIX RS/6000 32-bit and 64-bit	`xlc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybdb -lsybunic -o program`
HP HP-UX PA-RISC 32-bit and 64 bit	`cc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybdb -lsybunic -Wl,-E,+s -o program`
HP HP-UX Itanium 32-bit and 64-bit	`cc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybdb -lsybunic -Wl,-E,+s -o program`
Linux x86 32-bit	`cc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybdb -lsybunic -ldl -o program`
Linux POWER 32-bit and 64-bit	`xlc -q32 -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybdb -lsybunic -ldl -o program`
Linux x86-64 64-bit	`gcc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybdb64 -lsybunic64 -ldl -o program`

Table 2-3 shows commands for compiling and linking DB-Library applications on platforms that support shareable libraries (with dynamic drivers).

Table 2-3: Shareable compile-and-link commands for DB-Library

Platform	Command
Solaris SPARC 32-bit and 64-bit	`/opt/SUNWspro/bin/cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -R$SYBASE/$SYBASE_OCS.lib program.c -Bdynamic -lsybdb -o program`
Solaris x86-64 32-bit and 64-bit	`/opt/SunStudio10/SUNWspro/bin/cc -xtarget=opteron -xarch=amd64 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -R$SYBASE/$SYBASE_OCS.lib program.c -Bdynamic -lsybdb -o program`
IBM AIX RS/6000 32-bit and 64-bit	`xlc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bdynamic -lsybdb -o program`
HP HP-UX PA-RISC 32-bit and 64 bit	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,a,shared_archive -lsybdb -o program`
HP HP-UX Itanium 32-bit and 64-bit	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,a,shared_archive -lsybdb -o program`
Linux x86 32-bit	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bdynamic -lsybdb -ldl -o program`
Linux POWER 32-bit and 64-bit	`xlc -q32 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bdynamic -lsybdb -ldl -o program`
Linux x86-64 64-bit	`gcc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bdynamic -lsybdb64 -ldl -o program`

Performance considerations

Linking with shared libraries results in a smaller executable and is faster than linking with static libraries. However, executables that link with shared libraries may be slower at start-up time than those that link with static libraries. Unlike static libraries, shared libraries must be available at runtime.

Your individual site requirements determine the type of library that provides the best performance.

Header files

All DB-Library/C applications require these header files:

- *sybfront.h* – defines symbolic constants such as function return values, described in the *Open Client DB-Library/C Reference Manual*, and the exit values STDEXIT and ERREXIT. The *sybfront.h* file also includes type definitions for datatypes that can be used in program variable declaration.
- *sybdb.h* – contains additional definitions and typedefs, most of which are meant to be used only by the DB-Library/C routines. Use the contents of *sybdb.h* only as documented in the *Open Client DB-Library/C Reference Manual*.
- *syberror.h* – contains error severity values and should be included if the program refers to those values.

See the *Open Client DB-Library/C Reference Manual.*

Using DB-Library sample programs

Sample programs are included with DB-Library to demonstrate typical uses for DB-Library routines.

Some sample programs use the sample databases supplied with Adaptive Server. See the *Adaptive Server Enterprise Installation Guide* for information on installing the sample databases.

Purpose of the sample programs

The sample programs demonstrate specific DB-Library functionality. These programs are designed as guides for application programmers, not as DB-Library training aids. Read the descriptions at the top of each source file and examine the source code before you use the sample programs.

Note These simplified programs are not intended for use in a production environment. Production-quality programs require additional code to handle errors and special cases.

Location

The sample programs are located in *$SYBASE/$SYBASE_OCS/sample/dblibrary*.

This directory contains:

- Source code for the sample programs
- Data files for the samples
- The samples header file, *sybdbex.h*
- The *README* file containing instructions for building, executing, and testing the samples

Before compiling and running the sample programs, copy the contents of *$SYBASE/$SYBASE_OCS/sample/dblibrary* into a working directory, where you can experiment with the sample programs without affecting the integrity of the original files.

Header file

All of the sample programs reference the sample header file, *sybdbex.h*. The contents of *sybdbex.h* are as follows:

```
/*
 ** sybdbex.h
 **
 ** This is the header file that goes with the
 ** Sybase DB-Library sample programs.
 **
```

```
**
*/

#define USER               "sa"
#define PASSWORD           ""
#define LANGUAGE           "us_english"
#define SQLBUFLEN          255
#define ERR_CH             stderr
#define OUT_CH             stdout
extern void                error();
int CS_PUBLIC err_handler PROTOTYPE((
DBPROCESS   *dbproc,
int         severity,
int         dberr,
int         oserr,
char        *dberrstr,
char        *oserrstr));

int CS_PUBLIC msg_handler PROTOTYPE((
DBPROCESS   *dbproc,
DBINT       msgno,
int         msgstate,
int         severity,
char        *msgtext,
char        *srvname,
char        *procname,
int          line));
```

All of the samples except Example 5 contain these lines:

```
DBSETLUSER(login, USER);
DBSETLPWD(login, PASSWORD);
```

The changes you can make for the lines in *sybdbex.h* include:

- USER is defined in *sybdbex.h* as “sa.” Before running the sample programs, edit *sybdbex.h* to change “sa” to your server login name.
- PASSWORD is defined in *sybdbex.h* as a null (“ ”) string. Before running the sample programs, edit *sybdbex.h* to change “server_password” to your server password. Choose one of the following options for PASSWORD:
    - Change your server password to “server_password” while you are running the samples. This creates the possibility of a security breach, because while your password is set to this published value, an unauthorized person might take the opportunity to log in to the server as you. If this is a problem, choose one of the other options.

- In *sybdbex.h,* change the null (“ ”) string to your own server password. Use the operating system’s protection mechanisms to prevent others from accessing the header file while you are using it. When you are finished with the sample, edit the line so that it again says “server_password.”
- In the sample programs, delete the DBSETLPWD line entirely, and substitute your own code to prompt users for their server passwords. Because this code is platform-specific, Sybase does not supply it.

- If your server’s language is not U. S. English, edit the LANGUAGE line in *sybdbex.h* so that it is the same as the server’s. Example 12 is the only sample that references LANGUAGE.

Sample program summaries

These are the sample programs that are included with your software.

example1.c sample program

The *example1.c* sends two queries to Adaptive Server in a single command batch, binds the results, and prints the returned rows of data.

example2.c sample program

The *example2.c* inserts data from a file into a newly created table, selects the server rows, and binds and prints the results. This sample requires a file named *datafile* (supplied). It also assumes that you have create database permission in your login database.

example3.c sample program

The *example3.c* selects information from the titles table in the pubs2 database and prints it. The sample program illustrates binding of both aggregate and compute results.

Note To use this sample, you must be able to access to Adaptive Server and the pubs2 database.

example4.c sample program

The *example4.c* demonstrates row buffering. This program sends a query to Adaptive Server, buffers the returned rows, and allows you to interactively examine the rows.

example5.c sample program

The *example5.c* illustrates dbconvert, a DB-Library/C routine that handles data conversion.

example6.c sample program

The *example6.c* demonstrates browse-mode techniques. The sample program creates a table, inserts data into the table, and then updates the table using browse-mode routines. Browse mode is useful for applications that need to update data one row at a time.

Note *example6.c* requires a file named *datafile* (supplied). It creates the table alltypes in your default database.

example7.c sample program

The *example7.c* uses browse-mode techniques to determine the source of result columns from ad hoc queries. Determining the source of result columns is important because a browse-mode application can update only columns that are derived from a browsable table and are not the result of a SQL expression.

This sample program demonstrates how an application can determine which columns resulting from ad hoc queries can be updated using browse-mode techniques. It also prompts you for an ad hoc query. The results differ, depending on whether the select query includes the keywords for browse and whether the selected table can be browsed.

example8.c sample program

The *example8.c* sends a remote procedure call, prints the result rows from the call, and prints the parameters and status returned by the remote procedure.

This sample requires you to have created the stored procedure rpctest in your default database. The comments at the top of the *example8.c* source code specify the create procedure statement necessary for creating rpctest.

example9.c sample program

The *example9.c* generates a random image, inserts it into a table, then selects the image and compares it to the original:

1. insert all data into the row except the text or image value.
2. update the row, setting the value of the text or image to NULL. This step is necessary because a text or image column row that contains a null value includes a valid text pointer only if the null value was explicitly entered using the update statement.
3. select the row. You must specifically select the column that is to contain the text or image value. This step provides the application's DBPROCESS with correct text pointer and text timestamp information. The application should throw away the data returned by this select statement.
4. Call dbtxtptr to retrieve the text pointer from the DBPROCESS. dbtxtptr's *column* parameter is an integer that refers to the select performed in step 3. For example, if the select is:

   ```
   select date_column, integer_column, text_column
          from bigtable
   ```

   and text_column is the name of the text column, dbtxtptr requires the *column* parameter to be passed as 3.
5. Call dbtxtimestamp to retrieve the text timestamp from the DBPROCESS. dbtxtimestamp's column parameter refers to the select performed in step 3.
6. Write the text or image value to Adaptive Server. An application can either:
   - Write the value with a single call to dbwritetext, or
   - Write the value in chunks, using dbwritetext and dbmoretext.
7. If you intend the application to make another update to this text or image value, it may want to save the new text timestamp that is returned by Adaptive Server at the conclusion of a successful dbwritetext operation. Access the new text timestamp by using dbtxtsnewval, and store it for later retrieval using dbtxtsput.

Note To use this sample, you must be able to access Adaptive Server that contains the pubs2 database.

example10.c **sample program**

The *example10.c* prompts you for an author ID and the name of a file containing an image, reads the image from the file, and inserts a new row containing the author ID and the image into the pubs2 database table called au_pix. For general information on inserting text or image values into a database table, see example9.c.

Note To use this sample, you must be able to access Adaptive Server that contains the pubs2 database. The author ID must be in the form 000-00-0000. The *imagefile* file, provided with the sample code, contains an image.

example11.c **sample program**

The *example11.c* retrieves an image from the au_pix table in the pubs2 database. The author ID you enter determines which row the program selects. After retrieving the row, this sample copies the image contained in the pic field to a file you specify.

There are two ways to retrieve a text or image value from Adaptive Server:

- This sample selects the row containing the value and processes the row using dbnextrow. After dbnextrow is called, dbdata can be used to return a pointer to the returned image.
- The other method is to use dbreadtext with dbmoretext to read a text or image value in the form of a number of smaller chunks.

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

Note To use this sample, you must be able to access Adaptive Server and the pubs2 database.

example12.c **sample program**

The *example12.c* retrieves data from the pubs2 database and prints it using a us_english format.

Note To use this sample, you must be able to access Adaptive Server and the pubs2 database.

bulkcopy.c sample program

The *bulkcopy.c* uses the bulk-copy routines to copy data from a host file into a newly created table containing several Adaptive Server datatypes.

Note To use this sample, you must be able to access Adaptive Server. You must also have create database and create table permission.

---

twophase.c sample program

The *twophase.c* commit performs a simple update on two different servers. See the source code for the exact contents of the update. After you have run the sample, use isql on each server to determine whether the update actually took place.

This sample requires that you have Adaptive Server running on two different servers, named SERVICE and PRACTICE, each containing the pubs2 database. If your servers are named differently, replace SERVICE and PRACTICE in the source code with the actual names of your servers.

Before running the sample, make sure that your client can access both servers. See the *Open Client and Open Server Configuration Guide for UNIX*.

Note If the PRACTICE server is on a different machine than the SERVICE server, the PRACTICE server must be able to connect to the SERVICE query port.

CHAPTER 3 Open Server Server-Library/C

Use Open Server Server-Library/C to design servers that take advantage of the features of the client/server architecture. These Open Servers access data stored in non-Sybase database management systems, trigger external events, and respond to Open Client applications.

The client/server architecture divides the work of computing between "clients" and "servers":

- Clients make requests of servers and process the servers' responses.
- Servers respond to requests and return data, parameters, and status information to clients.

In this architecture, an Open Client application program is a client, using the services provided by Adaptive Server and Open Server. Using Server-Library, you can create a complete, standalone server.

See the *Open Server and SDK New Features for Windows, Linux, and UNIX* for a list of operating system platforms where the Open Server Server-Library/C is available.

General instructions

To run Open Server applications, including samples, you must:

- Be able to access Adaptive Server and the pubs2 sample database. See the *Adaptive Server Enterprise Installation Guide* for information on installing the pubs2 database.
- Set these environment variables, which are described in Appendix B, "Environment Variables":

    - SYBASE
    - SYBASE_OCS
    - DSQUERY and DSLISTEN
    - SYBPLATFORM
    - Platform-specific library path variable
- Be able to connect to an Adaptive Server. See the *Open Client and Open Server Configuration Guide for UNIX*.
- Set execute permission on the *sybopts.sh* file for the file's owner:

  ```
  chmod u+x sybopts.sh
  ```

Building a Server-Library executable

Use libraries, linking, and header files to build a Server-Library executable.

Libraries

Table 3-1 lists the libraries to include to take full advantage of all Server-Library capabilities. The first row in the table lists libraries that all platforms use. Subsequent rows list platform-specific libraries.

Table 3-1: Platform-specific libraries

Platform	Required libraries
All platforms	*libsybct* – Client-Library (Sybase) *libsybcs* – CS-Library (Sybase) *libsybtcl* – transport control layer (Sybase internal) *libsybcomn* – internal shared utility library (Sybase internal) *libsybintl* – internationalization support library (Sybase internal) *libsybunic* – Unicode-Library (Sybase internal) *libsybsrv* – Server-Library (Sybase) *libsybdb* – DB-Library (Sybase) *libm* – standard UNIX math library (system)
Solaris platforms	*libthread* – thread library (system) *libpthread* – thread library (system) *libsocket* – socket network library (system) *libnsl* – a network library (system) *libdl* – dynamic loader library (system)
HP HP-UX platforms	*libcl* – HP transport control layer (system) *libBSD* – the BSD library (system) *libc_r* – C reentrant library *libdld* – dynamic loader library (system)
IBM AIX platforms	*libc_r* – C reentrant library *libpthreads* – thread library (system)
Linux platforms	*libpthread* – thread library (system)

Compile-and-link line commands

The following tables list the general forms of the commands for compiling and linking Server-Library applications on Sybase-supported platforms running the UNIX operating system.

Table 3-2 shows commands for compiling and linking Server-Library applications using static libraries:

Table 3-2: Static compile-and-link commands for Server-Library

Platform	Command
Solaris SPARC 32-bit and 64-bit	`/opt/SUNWspro/bin/cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Bstatic -lsybsrv [ -lsybdb | -lsybct ] -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -Bdynamic  -lnsl -ldl -lm -lsocket -o program`

Platform	Command
Solaris x86-64 32-bit and 64-bit	`/opt/SunStudio10/SUNWspro/bin/cc -xtarget=opteron -xarch=amd64 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Bstatic -lsybsrv [ -lsybdb | -lsybct ] -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -Bdynamic  -lnsl -ldl -lm -lsocket -o program`
IBM AIX RS/6000 32-bit and 64-bit	`xlc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bstatic -lsybsrv [ -lsybdb | -lsybct ] -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -Wl,-Bdynamic -lm -o program`
HP HP-UX PA-RISC 32-bit and 64 bit	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-a,archive -lsybsrv [ -lsybdb | -lsybct ] -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -Wl,-a,shared_archive -lcl -lm -lBSD -Wl,-E,+s -o program`
HP HP-UX Itanium 32-bit and 64-bit	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-a,archive -lsybsrv [ -lsybdb | -lsybct ] -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -Wl,-a,shared_archive -lcl -lm -lBSD -Wl,-E,+s -o program`
Linux x86 32-bit	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bstatic -lsybsrv [ -lsybdb|-lsybct ] -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -Wl,-Bdynamic -ldl -lnsl -lm -o program`
Linux POWER 32-bit and 64-bit	`xlc -q32 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bstatic -lsybsrv -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -Wl,-Bdynamic -ldl -lnsl -lm -o program`
Linux x86-64 64-bit	`gcc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bstatic -lsybsrv64 -lsybct64 -lsybcs64 -lsybtcl64 -lsybcomn64 -lsybintl64 -lsybunic64 -Wl,-Bdynamic -ldl -lnsl -lm64 -o program`

Table 3-3 shows commands for compiling and linking Server-Library applications using debug libraries:

Table 3-3: Debug compile-and-link commands for Server-Library

Platform	Command
Solaris SPARC 32-bit and 64-bit	`/opt/SUNWspro/bin/cc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybsrv [-lsybdb | -lsybct] -lsybcs -lnsl -lm -lsocket -o program`
Solaris x86-64 32-bit and 64-bit	`/opt/SunStudio10/SUNWspro/bin/cc -xtarget=opteron -xarch=amd64 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybsrv [-lsybdb | -lsybct] -lsybcs -lnsl -lm -lsocket -o program`
IBM AIX RS/6000 32-bit and 64-bit	`xlc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybsrv [-lsybdb| -lsybct] -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lm -o program`
HP HP-UX PA-RISC 32-bit and 64 bit	`cc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybsrv [-lsybdb | -lsybct] -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lcl -lm -lBSD -o program`
HP HP-UX Itanium 32-bit and 64-bit	`cc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybsrv [-lsybdb | -lsybct] -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lcl -lm -lBSD -o program`
Linux x86 32-bit	`cc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -lsybsrv [-lsybdb|-lsybct] -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -ldl -lnsl -lm -o program`
Linux POWER 32-bit and 64-bit	`xlc -q32 -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -lsybsrv [-lsybdb|-lsybct] -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -ldl -lnsl -lm -o program`
Linux x86-64 64-bit	`gcc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -lsybsrv64 [-lsybdb64|-lsybct64] -lsybcs64 -lsybtcl64 -lsybcomn64 -lsybintl64 -lsybunic64 -ldl -lnsl -lm64 -o program`

Table 3-4 shows commands for compiling and linking Server-Library applications using shareable libraries (with dynamic drivers):

Table 3-4: Shareable compile-and-link commands for Server-Library

Platform	Command
Solaris SPARC 32-bit and 64-bit	`/opt/SUNWspro/bin/cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -R$SYBASE/lib program.c -Bdynamic -lsybsrv [ -lsybdb | -lsybct ] -lsybcs -lnsl -ldl -lm -lsocket -o program`
Solaris x86-64 32-bit and 64-bit	`/opt/SunStudio10/SUNWspro/bin/cc -xtarget=opteron -xarch=amd64 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -R$SYBASE/lib program.c -Bdynamic -lsybsrv [ -lsybdb | -lsybct ] -lsybcs -lnsl -ldl -lm -lsocket -o program`
IBM AIX RS/6000 32-bit and 64-bit	`xlc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -R$SYBASE/lib program.c -Wl,-Bdynamic -lsybsrv [-lsybdb| -lsybct] -lsybcs -lm -o program`
HP HP-UX PA-RISC 32-bit and 64 bit	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,a,shared_archive -lsybsrv [ -lsybdb | -lsybct ] -lsybcs -lcl -lm -lBSD -o program`
HP HP-UX Itanium 32-bit and 64-bit	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,a,shared_archive -lsybsrv [ -lsybdb | -lsybct ] -lsybcs -lcl -lm -lBSD -o program`
Linux x86 32-bit	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bdynamic -lsybsrv [ -lsybdb | -lsybct ] -lsybcs -ldl -lnsl -lm -o program`
Linux POWER 32-bit and 64-bit	`xlc -q32 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bdynamic -lsybsrv [ -lsybdb | -lsybct ] -lsybcs -ldl -lnsl -lm -o program`
Linux x86-64 64-bit	`gcc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bdynamic -lsybsrv64 [ -lsybdb64 | -lsybct64 ] -lsybcs64 -ldl -lnsl -lm64 -o program`

Note The Open Server program can use Client-Library or DB-Library routines. The bracketed information after -lsybsrv in the above lines means that you can choose either -lsybdb for DB-Library or -lsybct for Client-Library.

Table 3-5 shows commands for compiling and linking Server-Library applications with libraries to take advantage of thread-safe support:

Table 3-5: Thread-safe compile-and-link commands for Server-Library

Platform	Command
Solaris SPARC 32-bit and 64-bit	`/opt/SUNWspro/bin/cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -D_REENTRANT -lsybsrv_r -lsybct_r -lsybcs_r -lsybtcl_r -lsybcomn_r -lsybintl_r -lnsl -ldl -lpthread -lthread -lm -lsocket -o program`
Solaris x86-64 32-bit and 64-bit	`/opt/SunStudio10/SUNWspro/bin/cc -xtarget=opteron -xarch=amd64 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -D_REENTRANT -lsybsrv_r -lsybct_r -lsybcs_r -lsybtcl_r -lsybcomn_r -lsybintl_r -lnsl -ldl -lpthread -lthread -lm -lsocket -o program`
IBM AIX RS/6000 32-bit and 64-bit	`xlc_r -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -D_THREAD_SAFE -lsybsrv_r -lsybct_r -lsybcs_r -lsybtcl_r -lsybcomn_r -lsybintl_r -lpthread -lm -o program`
HP HP-UX PA-RISC 32-bit and 64 bit	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -D_THREAD_SAFE -D_REENTRANT -Ae -lsybsrv_r -lsybct_r -lsybcs_r -lsybtcl_r -lsybcomn_r -lsybintl_r -lcl -lm -lBSD -lpthread -ldld -o program`
HP HP-UX Itanium 32-bit and 64-bit	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -D_THREAD_SAFE -D_REENTRANT -Ae -lsybsrv_r -lsybct_r -lsybcs_r -lsybtcl_r -lsybcomn_r -lsybintl_r -lcl -lm -lBSD -lpthread -ldld -o program`
Linux x86 32-bit	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib` *`-lsybsrv_r -lsybct_r`* `-lsybcs_r -lsybtcl_r -lsybcomn_r -lsybintl_r -ldl -lpthread -lnsl -lm -o program`
Linux POWER 32-bit and 64-bit	`xlc_r -q32 -g -D_REENTRANT -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib` *`-lsybsrv_r -lsybct_r`* `-lsybcs_r -lsybtcl_r -lsybcomn_r -lsybintl_r -ldl -lpthread -lnsl -lm -o program`

Platform	Command
Linux x86-64 64-bit	`gcc_r -q32 -g -D_REENTRANT -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -lsybsrv64_r -lsybct64_r -lsybcs64_r -lsybtcl64_r -lsybcomn64_r -lsybintl64_r -ldl -lpthread -lnsl -lm64 -o program`

Kerberos support

Server-Library supports Kerberos security features for applications that need a high level of security when communicating over a network. By installing the required Kerberos software and performing the appropriate configuration tasks, your Server-Library applications can take advantage of these Kerberos security features:

- Network authentication
- Mutual authentication
- Out-of-sequence authentication
- Replay detection
- Confidentiality
- Integrity

Table 3-6: Required tasks for Kerberos support

Tasks	**For more information**
Install the following Kerberos software on your system. Be sure that the *GSS* library support is available as a shared library.	See your Kerberos documentation and see the *Open Client and Open Server Configuration Guide for UNIX*.
Extract keys for the desired server principal(s) into a key table file using the Kerberos utility called kadmin.	See your Kerberos documentation.
Configure the security section of the *libtcl.cfg* configuration file.	See the *Open Client and Open Server Configuration Guide for UNIX*.
Link your Client-Library application with the Sybase reentrant libraries.	See “Kerberos support” on page 49.
• For CyberSafe Kerberos: • Set the CSFC5CCNAME environment variable to the credential cache directory location. • Set the CSFC5KTNAME variable to the path of the key table file if other than the default key table file. • For MIT Kerberos • Set the KRB5CCNAME environment variable to the credential cache file location. • Set the KRB5_KTNAME variable to the path of the key table file if other than the default key table file.	See your Kerberos documentation. Default credential cache directory location varies by platform. • For CyberSafe Trust Broker, the default key table file is */krb5/v5srvtab*. • For MIT Kerberos, the default key table file is */etc/krb5.keytab*.
Use srv_props to set the server principal name if it is different from the server name passed to srv_init.	See the *Open Server Server-Library/C Reference Manual*.

Note To avoid compromising security, Sybase suggests that the *key table* files be owned by the user ID that runs Open Server, and that all other users be restricted from accessing this file. Sybase also suggests that you run each Open Server using a unique user id that is not used by interactive processes.

Bulk copy routines

If you plan to use bulk copy routines, link in the bulk copy library. Add `-llsybblk` before `-lsybsrv` in the link command line.

See the *Open Client and Open Server Common Libraries Reference Manual*.

Performance considerations

Linking with shared libraries results in a smaller executable and takes less time than linking with static libraries. However, executables that link with shared libraries may have a slower start-up time than those that link with static libraries. Unlike static libraries, shared libraries must be available at runtime.

The type of library that provides the best performance is determined by the individual requirements of your site.

Header files

Include the *ospublic.h* header file in all Open Server application source files. Other necessary header files are nested in *ospublic.h*. If Bulk-Library is used, include *bkpublic.h* in addition to *ospublic.h*.

See the *Open Server Server-Library/C Reference Manual* for more information on header files.

Using Server-Library sample programs

This section contains information about the sample programs that are included with Server-Library.

The sample programs demonstrate typical uses for Server-Library routines in C programs. The sample programs are servers, and therefore require entries in the *interfaces* file or entries in a network directory service to describe their machines and network addresses. See the *Open Client and Open Server Configuration Guide for UNIX* for information about configuring directory services.

Purpose of the sample programs

The sample programs demonstrate specific Open Server functionality. With the exception of ctos, these programs are designed as guides for application programmers, not as Open Server training aids. Read the descriptions at the top of each source file and examine the source code before attempting to use the sample programs.

Note These simplified programs are not intended for use in a production environment. Production-quality programs require additional code to handle errors and special cases.

Check the individual sample programs to see which trace flags can be used with them. See the *README* file for complete instructions on running the sample programs.

Location

The sample programs are located in *$SYBASE/$SYBASE_OCS/sample/srvlibrary*, which includes:

- Source code for the sample programs.
- The *makefile* provided to build the samples. Use the samples as a starting point for your own Server-Library applications.
- The samples header file, *ossample.h.*
- A srv_connect event handler.
- Error handlers.
- The *README* file containing instructions for building, executing, and testing the samples.

Sample program summaries

These are the sample programs that are included with your software.

Note For single-threaded and multithreaded samples, a client uses the stop_serv registered procedure to stop the samples.

---

ctos.c sample program

The *ctos.c* is an Open Server gateway application that uses Server-Library calls and Client-Library calls. It accepts commands from a client and passes them to a remote Adaptive Server. Then, it retrieves the results from the remote server and passes them to the client. *ctos.c* processes a variety of client commands:

- Bulk-copy commands
- Cursor commands
- Scrollable cursor commands
- Dynamic SQL commands
- Language commands
- Option commands
- Remote procedure calls (RPCs)

In addition, ctos.c responds to attention requests from a client by calling the srv_attention event handler. It includes an event handler routine to process each type of client command.

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

dynlisten.c sample program

The *dynlisten.c* sample demonstrates how a running Open Server can start a new listener on a wild-carded IP port number and retrieve the final address information in a manner that can then be used to redirect logins or migrate connections.

exfds.c sample program

The *exfds.c* demonstrates how an Open Server application can service external file descriptors without blocking the entire Open Server process. This program:

- Verifies that the current platform supports srv_poll, using the srv_capability routine
- Opens two UNIX pipes
- Spawns two service threads, srv_poll and srv_stop, using the srv_spawn routine

The two service threads implement a simple command/response protocol by writing messages on the UNIX pipes. srv_poll allows Open Server to reschedule the service thread while waiting for a message. Information is written to *srv.log* to monitor the progress. The Open Server performs the command/response protocol the number of times specified in the source code, then queues a SRV_STOP event.

This sample does not require a client application. Check the *srv.log* file for messages to determine if it has started correctly.

fullpass.c sample program

The *fullpass.c* is an Open Server gateway application that uses the Sybase Tabular Data Stream™ (TDS) passthrough mode. See “Passthrough Mode” in Chapter 2 of the *Open Server Server-Library/C Reference Manual*.

The event handler routine receives client requests through srv_recvpassthru and forwards this information to an Adaptive Server using the ct_sendpassthru routine. After the entire client command has been forwarded to the remote server, the event handler reads results from the remote server through ct_recvpassthru and returns them to the client using srv_sendpassthru.

The application also includes a SRV_CONNECT event handler. This handler uses srv_getloginfo and ct_setloginfo to forward client connection information to the remote server. It then uses ct_getloginfo and srv_setloginfo to return connection acknowledgment information to the client. All Open Server applications that use TDS passthrough mode must include these calls in their SRV_CONNECT event handler.

halang.c sample program

halang.c demonstrates how to set the HA Session ID so that an Open Client can reconnect to Open Server after a failover. Use this sample program to implement HA functionality in Open Server.

To run, execute halang *your_open_server_name* -s *failover_server &*

Note Open Server application generates the HA Session ID.

---

intlchar.c **sample program**

The *intlchar.c* demonstrates Open Server handling of national languages and character sets. It initializes values for the Open Server application's national language and character set, and then changes these values in response to client requests.

Client requests come in the form of option commands and language commands. *intlchar.c* installs SRV_OPTION and SRV_LANGUAGE event handlers, as well as a SRV_CONNECT handler.

lang.c **sample program**

The *lang.c* demonstrates how to use a srv_language event handler. The event handler responds to client language commands with an informational message, which it sends to the client using the srv_sendinfo routine. This program also contains a srv_connect event handler and error handlers.

For more information on processing language commands, see “Language Calls” in the *Open Server Server-Library/C Reference Manual*.

multthrd.c **sample program**

The *multthrd.c* illustrates a number of Open Server multithreaded programming features, including:

- Creation of a service thread using srv_spawn
- Interthread communication between client connection threads and the service thread through message queues (using srv_getmsgq and srv_putmsgq)
- Sleep and wake-up mechanisms (using srv_sleep and srv_wakeup)
- The use of a callback routine (using srv_callback) to report scheduling information

A service thread logs all the language queries received by this Open Server application.

In the application’s language handler, the client thread reads the query from a client and sends a message to the service thread, known as the “logger,” with the query as the message data. Then, the client thread waits (srv_sleep). When the service thread gets the message, it wakes up the client thread (srv_wakeup). The logger thus continuously loops, waiting for messages. When it receives a message, it prints the contents of the query to a file and wakes up the sender.

The logger and client threads install SRV_C_RESUME, SRV_C_SUSPEND, SRV_C_TIMESLICE, and SRV_C_EXIT callback handlers to print scheduling information. The *multthrd.c* program installs a SRV_START handler, a SRV_LANGUAGE handler, a SRV_CONNECT handler, and callback handlers.

osintro.c sample program

The *osintro.c* demonstrates the basic components of an Open Server application. It has no event handlers installed.

mqueue.c sample program

The *mqueue.c* sample demonstrates the use of Open Server message queues using the srv_createmsgq(), srv_putmsgq(), and srv_getmsgq() API functions. Any language command sent to the server is stored in a message queue by the language handler. A service thread logger reads messages from the queue, displays them to stdout and stores them in the log.

paramreader.c sample program

The *paramreader.c* sample program demonstrates a simple standalone Open Server application. It installs a SRV_RPC and SRV_DYNAMIC event handlers to display incoming (LOB) parameters that are received from a client application. Client examples *lobrpc.c* and *lobdynamic.c* are provided to send such parameters. These samples are mainly provided to show how to send and receive TEXT, IMAGE, or UNITEXT type parameters with RPC's or Dynamic SQL.

redirect.c sample program

redirect.c is a simple Open Server application that causes an Open Client to log into a different server than was originally specified in the interfaces file. It only works with 15.0 or later clients.

To run, execute redirect *your_open_server_name* -s *alternate_server_to_use*

Note The server directed to can be based on information within the login packet (such as the application name or server name field). The client restarts its login attempt with using the connection information provided by the redirect server.

regproc.c sample program

The *regproc.c* demonstrates the use of registered procedures in Open Server. The application registers several procedures at start-up time, and then waits for client commands. No Open Server event handlers are installed.

Clients send RPC commands to execute the registered procedures defined in *regproc.c*.

Several additional client programs are provided for use with *regproc.c*:

- *version.c* – executes a registered procedure (rp_version) that returns the version of the Open Server.
- *dbwait.c* – implemented with DB-Library, registers with the Open Server to be notified when rp_version is executed.
- *ctwait.c* – implemented with Client-Library, registers with the Open Server to be notified when rp_version is executed.

secsrv.c sample program

The *secsrv.c* demonstrates how Open Server uses network-based security services. The connection handler in this sample program retrieves the security properties of the client thread and sends messages to the client that describe which security services are active for the session.

See the *Open Client and Open Server Configuration Guide for UNIX*.

sendrpc.c sample program

The *sendrpc.c* sample demonstrates sending a simple RPC command to an Adaptive Server or to an Open Server application and handling the returned results. The sample just takes simple RPC commands without parameters. For example, sp_who or ctos_shutdown.

sigalarm.c sample program

The *sigalarm.c* demonstrates how an Open Server application can use a UNIX SIGALARM signal to schedule periodic events. Specifically, *sigalarm.c*:

- Spawns, using srv_spawn, a service thread that sleeps until an alarm wakes it up. Each time the service thread is awakened, it writes a message to the Open Server log file using the srv_log routine.

- Installs a SIGALARM handler, using the srv_signal routine, that wakes up a sleeping service thread each time the SIGALARM handler is called. *sigalarm.c* requests that a SIGALARM be delivered at a particular interval, using the UNIX alarm call.

This sample does not require a client application. Check the *srv.log* file for messages to determine if it has started correctly.

timedsleep.c sample program

The *timedsleep.c* sample demonstrates the use of the srv_timedsleep(), srv_createmutex(), srv_lockmutex(), and srv_unlockmutex() API functions. Use the sample to play the Rock-Paper-Scissors game with two isql connections.

Note The srv_timedsleep() API function and this sample program are only available with the threaded libraries.

updtext.c sample program

The *updtext.c* sample program is used in conjunction with *uctext ctlibrary* sample. It installs a SRV_LANGUAGE event handler and responds to client commands with an informational message that contains the language command received.

Additionally, the sample also installs a bulk handler that sends a message to the client indicating the text received. This is to demonstrate the updatetext functionality. Since the server can be queried to see if the feature is supported, an rpc_handler is installed. The rpc_handler checks for the sp_mda rpc. This rpc responds to the client with metadata about the server (specifically that partial text update is supported).

CHAPTER 4 Open Client Embedded SQL/C

Embedded SQL is a superset of Transact-SQL® that lets you embed Transact-SQL statements in application programs written in languages like C. Embedded SQL includes all Transact-SQL statements, and the extensions needed to use Transact-SQL in an application program.

Embedded SQL provides a simple way to retrieve, insert, or modify data stored in any Adaptive Server database.

See the *Open Server and SDK New Features for Windows, Linux, and UNIX* for a list of operating system platforms where the Open Client Embedded SQL/C is available.

General instructions

To run Embedded SQL/C applications, including the sample programs, you must:

- Set these environment variables, which are described in Appendix B, “Environment Variables”:
  - SYBASE
  - SYBASE_OCS
  - SYBPLATFORM
  - Platform-specific library path variable
- Be able to access an Adaptive Server on which the pubs2 sample database is installed. See the *Adaptive Server Enterprise Installation Guide* for information about installing the pubs2 database.

- Set execute permission on the *sybopts.sh* file for the file's owner:

```
chmod u+x sybopts.sh
```

- If you have not already done so, include the current directory in the search path:

```
setenv PATH .:$PATH
```

Building an Embedded SQL/C executable

To build an executable program from an Embedded SQL application:

1. Precompile the application.
2. Compile the C source code generated by the precompiler, and link your application to any necessary files and libraries.
3. Load any precompiler-generated stored procedures.

Precompiling the application

The format of the command to precompile a source program is:

```
cpre
   [-Ccompiler]
   [-Ddatabase_name]
   [-Ffips_level]
   [-G[isql_file_name]]
   [-H]
   [-Iinclude_path_name]
   [-Jcharset_locale_name]
   [-Ksyntax_level]
   [-L[listing_file_name]]
   [-Ninterface_file_name]
   [-Otarget_file_name]
   [-Ppassword]
   [-Sserver_name]
   [-Ttag_id]
   [-Uuser_id]
   [-Vversion_number]
   [-Zlanguage_locale_name]
   [@options_file]...
   [-a] [-b] [-c] [-d] [-e] [-f] [-h] [-l] [-m] [-p] [-r] [-s] [-u] [-v] [-w] [-x] [-y]
   filename[.ext]
```

filename is the name of the Embedded SQL/C source file. The default extension for *filename* is ".cp". cpre generates an output file with a ".c" extension.

Note The precompilers for 64-bit applications are cpre64 and cpre_r64. The cpre64 precompiler is the non-reentrant precompiler, and the cpre_r64 is the reentrant version. You can use these precompilers on any 64-bit platform supported for Open Client and Open Server.

Some of the options are switches that activate features of the precompiler. For example, an option can generate a stored procedure. By default, these features are off, you can turn them on by including the option on the cpre command line. Other statement qualifiers specify values for the preprocessor—a password, for example. Enter the value after the option (with or without intervening spaces).

If you enter an invalid option, the precompiler lists the options that are available.

See Appendix A, "Utility Commands Reference," for detailed descriptions of precompiler options.

Compiling and linking the application

Use libraries, linking, and header files to compile and link an Embedded/SQL C application.

Client-Library and Server-Library support dynamic loading of Net-Library™, directory, and security drivers. You do not need to explicitly link these Sybase object files with your applications:

- Net-Library drivers
- Directory drivers
- Security drivers

The following tables list the general forms of the commands for compiling and linking Embedded SQL/C applications on Sybase supported platforms running on UNIX.

Table 4-1 shows the commands for compiling and linking Embedded SQL/C applications using static libraries.

Table 4-1: Static compile-and-link commands for Embedded SQL/C

Platform	Command
Solaris SPARC 32-bit and 64-bit	`/opt/SUNWspro/bin/cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib` *`gen_program.c`* `$SYBASE/$SYBASE_OCS/include/sybesql.c -Bstatic -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -Bdynamic -lnsl -ldl -lm -lsocket -o program`
Solaris x86-64 32-bit and 64-bit	`/opt/SunStudio10/SUNWspro/bin/cc -xtarget=opteron -xarch=amd64 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib` *`gen_program.c`* `$SYBASE/$SYBASE_OCS/include/sybesql.c -Bstatic -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -Bdynamic -lnsl -ldl -lm -lsocket -o program`
IBM AIX RS/6000 32-bit and 64-bit	`xlc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -Wl,-Bstatic -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lm -o program`
HP HP-UX PA-RISC 32-bit and 64 bit	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -Wl,-a,archive -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -Wl,-a,default -lcl -lm -lBSD -ldld -Wl, -E, +s -o program`
HP HP-UX Itanium 32-bit and 64-bit	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -Wl,-a,archive -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -Wl,-a,default -lcl -lm -lBSD -ldld -Wl, -E, +s -o program`
Linux x86 32-bit	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -Wl,-Bstatic -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -Wl,-Bdynamic -ldl -lnsl -lm -o program`
Linux POWER 32-bit and 64-bit	`xlc -q32 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -Wl,-Bstatic -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -Wl,-Bdynamic -ldl -lnsl -lm -o program`

Platform	Command
Linux x86-64 64-bit	`gcc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -Wl, -Bstatic -lsybct64 -lsybcs64 -lsybtcl64 -lsybcomn64 -lsybintl64 -lsybunic64 -Wl, -Bdynamic -ldl -lnsl -lm64 -o program`

Table 4-2 shows the commands for compiling and linking Embedded SQL/C applications using debug libraries.

Table 4-2: Debug compile-and-link commands for Embedded SQL/C

Platform	Command
Solaris SPARC 32-bit and 64-bit	`/opt/SUNWspro/bin/cc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lnsl -ldl -lm -lsocket -o program`
Solaris x86-64 32-bit and 64-bit	`/opt/SunStudio10/SUNWspro/bin/cc -xtarget=opteron -xarch=amd64 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lnsl -ldl -lm -lsocket -o program`
IBM AIX RS/6000 32-bit and 64-bit	`xlc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lm -o program`
HP HP-UX PA-RISC 32-bit and 64 bit	`cc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib` *`gen_program.c`* `$SYBASE/$SYBASE_OCS/include/sybesql.c -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lcl -lm -lBSD -ldld -o program`
HP HP-UX Itanium 32-bit and 64-bit	`cc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib` *`gen_program.c`* `$SYBASE/$SYBASE_OCS/include/sybesql.c -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lcl -lm -lBSD -ldld -o program`
Linux x86 32-bit	`cc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -ldl -lnsl -lm -o program`
Linux POWER 32-bit and 64-bit	`xlc -q32 -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -ldl -lnsl -lm -o program`
Linux x86-64 64-bit	`gcc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -lsybct64 -lsybcs64 -lsybtcl64 -lsybcomn64 -lsybintl64 -lsybunic64 -ldl -lnsl -lm64 -o program`

Table 4-3 shows the commands for compiling and linking Embedded SQL/C applications using shareable libraries (with dynamic drivers).

Table 4-3: Shareable compile-and-link commands for Embedded SQL/C

Platform	Command
Solaris SPARC 32-bit and 64-bit	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -Bdynamic -lsybct -lsybcs -lnsl -ldl -lm -lsocket -o program`
Solaris x86-64 32-bit and 64-bit	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -Bdynamic -lsybct -lsybcs -lnsl -ldl -lm -lsocket -o program`
IBM AIX RS/6000 32-bit and 64-bit	`xlc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -Wl,-Bdynamic -lsybct -lsybcs -lm -o program`
HP HP-UX PA-RISC 32-bit and 64 bit	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -Wl,a,shared_archive -lsybct -lsybcs -lcl -lm -lBSD -o program`
HP HP-UX Itanium 32-bit and 64-bit	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -Wl,a,shared_archive -lsybct -lsybcs -lcl -lm -lBSD -o program`
Linux x86 32-bit	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -Wl,-Bdynamic -lsybct -lsybcs -ldl -lnsl -lm -o program`
Linux POWER 32-bit and 64-bit	`xlc -q32 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -Wl,-Bdynamic -lsybct -lsybcs -ldl -lnsl -lm -o program`
Linux x86-64 64-bit	`gcc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib gen_program.c $SYBASE/$SYBASE_OCS/include/sybesql.c -Wl,-Bdynamic -lsybct64 -lsybcs64 -ldl -lnsl -lm64 -o program`

Note The object produced by compiling the *sybesql.c* file contains utility routines that are used by Embedded SQL/C applications. You must link *sybesql.o* with every application for the application to work properly.

- The link line for an Embedded SQL/C application is identical to that used for a Client-Library application. In the link line, *gen_program.c* is the generated C file from cpre.
- -lsybct represents the linker option to link in the Open Client libraries that your code calls. In addition to -lsybct, you can also specify any or all of the following linker options, in the order shown:

For nonthreaded applications	For threaded applications
-lsybsrv (for Server-Library routines)	-lsybsrv_r (for Server-Library routines)
-lsybblk (for Bulk-Library routines)	-lsybblk_r (for Bulk-Library routines)
-lsybct (for Client-Library routines)	-lsybct_r (for Client-Library routines)

- To build a 64-bit C application, use the -DSYB_LP64 compiler option to ensure that the C compiler generates the correct code. See the sybopts.sh script available in *$SYBASE/$SYBASE_OCS/sample/esqlc*.

For HP HP-UX system users:

- The option -Wl,-a,archive causes the linker to statically link the Sybase libraries. Without it, shared versions of the Sybase libraries are used. In this case, the SH_LIB_PATH environment variable must include *$SYBASE/$SYBASE_OCS/lib* at runtime, and the application user must have read and execute permission on the libraries in *$SYBASE/$SYBASE_OCS/lib*.
- HP HP-UX does not use the SH_LIB_PATH environment variable at runtime unless the application is linked with the +s linker option. You must use the +s linker options so that the system can find Sybase libraries at runtime. -E is required to prevent undefined-symbol errors when driver libraries are loaded at runtime. See the HP-UX ld.

Additional considerations

Consider the following performance and alignment issues when compiling and linking your application.

Performance

Linking with shared libraries results in a smaller executable and takes less time than linking with static libraries. However, executables linked with shared libraries may have a slower start-up time than those linked with static libraries. Also, unlike static libraries, the shared libraries must be available at runtime.

The type of library that provides the best performance depends on the individual requirements of your site.

Data alignment on a 64-bit architecture

When building a 64-bit application, your data structure must be aligned on an 8-byte boundary, that is, to memory addresses that are multiples of 8 bytes. Similarly, the data structure of 32-bit applications must be aligned on a 4-byte boundary.

Loading stored procedures

If you use the precompiler -G flag to generate stored procedures, use isql to load the stored procedures into Adaptive Server before you execute the program. The format of the isql command to execute a generated script is:

```
isql -Uuserid -Ppassword < program.sql
```

where the -U and -P flags specify the user ID and password to log in to Adaptive Server.

See Appendix A, "Utility Commands Reference," for complete information about isql.

Using Embedded SQL/C sample programs

The Embedded SQL/C precompiler has sample programs that demonstrate typical Embedded SQL/C applications.

Purpose of the sample programs

The sample programs demonstrate specific Embedded SQL/C functions. These programs are designed as guides for application programmers, not as Embedded SQL/C training aids. Read the descriptions at the top of each source file and examine the source code before using the sample programs.

See the *README* file for complete instructions on running the sample programs.

Note These simplified programs are not intended for use in a production environment. Production-quality programs require additional code to handle errors and special cases.

Location

The sample programs are located in *$SYBASE/$SYBASE_OCS/sample/esqlc*.

This directory includes:

- Source code for the sample programs.
- The *makefile* provided to build the samples. Use the *makefile* as a starting point for your own Embedded SQL applications.
- The samples header file, *sybsqlex.h*.
- The *README* file, containing instructions for building, executing, and testing the samples.

Set execute permission on the *sybopts.sh* file for the file's owner:

```
chmod u+x sybopts.sh
```

Before compiling and running the sample programs, copy the contents of *$SYBASE/$SYBASE_OCS/sample/esqlc* into a working directory, where you can experiment with the sample programs without affecting the integrity of the original files.

Header file

Before you precompile the programs, edit the sample header file as described below, and replace the user name and password with values that are valid for your Adaptive Server. Comments in the programs show where you should make the changes.

All of the sample programs reference the sample header file, *sybsqlex.h*. The contents of *sybsqlex.h* are as follows:

```
/*************************************************
 *                                               *
 *  sybsqlex.h - header file for Embedded SQL/C  *
 *examples                                       *
 *                                               *
 *************************************************/

 #define USER     "username"
 #define PASSWORD "password"
 #define ERREXIT -1
 #define STDEXIT  0
```

All of the samples contain this line:

```
#include "sybsqlex.h"
```

USER and PASSWORD are defined in *sybsqlex.h* as user name and password. Before running the sample programs, edit *sybsqlex.h* to change the user name to your Adaptive Server login name and “password” to your Adaptive Server password.

example1.cp sample program

The *example1.cp* shows how to use cursors in an interactive query program. The program:

- Displays a list of book types; user selects one type
- Displays all titles in the selected book type; prompts for a title ID
- Displays detailed information about the selected title and continues prompting for title IDs
- Exits when Return is pressed at a prompt

example2.cp sample program

The *example2.cp* demonstrates updating a row through a cursor. The program:

- Displays the columns in the authors table, row by row.
- Lets the user update author information in all but the au_id column. If the user presses Return for column information, that column's data remains unchanged.
- Requires the user to confirm the update before sending the data to Adaptive Server.

exampleHA.cp sample program

The *exampleHA.cp* shows how you can use Embedded SQL/C code with high availability (HA) Failover. The program is similar to *example1.cp*, with the addition of failover processing. Error handlers are used to detect and handle failover.

uni_example1.cp sample program

The *uni_example1.cp* shows how to use cursors to guide an interactive query of the titles table. The program is similar to *example1.cp*, with the addition of displaying unichar/univarchar columns. The program:

- Binds the character datatype to the unichar/univarchar column.
- Accesses unichar/univarchar data from the server, and displays in the character format of the client's character set.

uni_example2.cp sample program

The *uni_example2.cp* shows how you can use cursors to display and edit rows of a table. The program is similar to *example2.cp*, with the addition of displaying unichar/univarchar columns. The program:

- Binds the character datatype to the unichar/univarchar column.
- Accesses unichar/univarchar data from the server, and displays in the character format of the client's character set.

CHAPTER 5

Open Client Embedded SQL/COBOL

Embedded SQL is a superset of Transact-SQL that lets you embed Transact-SQL statements in application programs written in a language like COBOL. Embedded SQL includes all Transact-SQL statements, and the extensions needed to use Transact-SQL in an application.

Embedded SQL/COBOL provides a simple way to retrieve, insert, or modify data stored in any Adaptive Server database.

See the *Open Server and SDK New Features for Windows, Linux, and UNIX* for a list of operating system platforms where the Open Client Embedded SQL/COBOL is available.

General instructions

To run Embedded SQL/COBOL applications, including the sample programs, you must:

- Be able to access an Adaptive Server on which the pubs2 sample database is installed. See the *Adaptive Server Enterprise Installation Guide*.
- Set these environment variables, which are described in Appendix B, "Environment Variables":
    - SYBASE
    - SYBASE_OCS
    - COBDIR

- PATH
- SYBPLATFORM
- Platform-specific library path variable

Building an Embedded SQL/COBOL executable

Use libraries, linking, and the header files to build an Embedded SQL/COBOL executable.

Libraries

Table 5-1 lists libraries to include to take full advantage of all Embedded SQL/COBOL capabilities. The first row in the table lists libraries that all platforms can use. Subsequent rows list libraries specific for each platform:

Table 5-1: Platform-specific libraries for Embedded SQL/COBOL

Platform	Supported libraries
All platforms	*libsybcobct* – COBOL interface to Client-Library and CS-Library (Sybase) *libsybct* – Client-Library (Sybase) *libsybcs* – CS-Library (Sybase) *libsybunic* – Unicode-Library (Sybase internal) *libsybcomn* – an internal shared-utility library (Sybase internal) *libsybintl* – internationalization support library (Sybase internal) *libsybtcl* – transport control layer (Sybase internal) *libm* – standard UNIX math library (system)
Solaris platforms	*libpthread* – thread library (system) *libsocket* – socket network library (system) *libnsl* – a network library (system) *libdl* – dynamic loader library (system) *libthread* – native thread library (system)
HP HP-UX PA-RISC	*libsybcobct.sl* – shared dynamic 32-bit veneer layer library *libsybcobct_r.sl* – shared dynamic 32-bit veneer layer library (reentrant version) *libsybcobct64.sl* – shared dynamic 64-bit veneer layer library *libsybcobct_r64.sl* – shared dynamic 64-bit veneer layer library (reentrant version)
HP HP-UX Itanium	*libcl* – HP transport control layer (system) *libBSD* – the BSD library (system) *libc_r* – C reentrant library *libdld* – dynamic loader library (system)
IBM AIX RS/6000	*libc_r* – C reentrant library *libpthread* – thread library (system)
Linux x86 32-bit	*libsybcobct* – COBOL interface to Client-Library and CS-Library (Sybase) *libsybct* – Client-Library (Sybase) *libsybcs* – CS-Library (Sybase) *libsybtcl* – transport control layer (Sybase internal) *libsybcomn* – an internal shared-utility library (Sybase internal) *libsybintl* – internationalization support library (Sybase internal) *libsybunic* – Unicode-Library (Sybase internal) *libdl* – dynamic loader library (system) *libnsl* – a network library (system) *libm* – standard UNIX math library (system)

Platform	Supported libraries
All other platforms that support ESQL/ COBOL	*libsybcobct.so* – shared dynamic 32-bit veneer layer library *libsybcobct_r.so* – shared dynamic 32-bit veneer layer library (reentrant version) *libsybcobct64.so* – shared dynamic 64-bit veneer layer library *libsybcobct_r64.so* – shared dynamic 64-bit veneer layer library (reentrant version)

There are three basic steps to building an executable program from an Embedded SQL/COBOL application:

1. Precompile the application.
2. Compile and link the COBOL source code generated by the precompiler.
3. Load any precompiler-generated stored procedures.

Precompiling the application

The format of the command to precompile an Embedded SQL/COBOL source program is:

```
cobpre
    [-Ccompiler]
    [-Ddatabase_name]
    [-Ffips_level]
    [-G[isql_file_name]]
    [-Iinclude_path_name]
    [-Jcharset_locale_name]
    [-Ksyntax_level]
    [-L[listing_file_name]]
    [-Ninterface_file_name]
    [-Otarget_file_name]
    [-Ppassword]
    [-Sserver_name]
    [-Ttag_id]
    [-Uuser_id]
    [-Vversion_number]
    [-Zlanguage_locale_name]
    [@ options_file]
    [-a] [-b] [-c] [-d] [-e] [-f] [-l] [-m] [-r] [-s] [-u] [-v] [-w] [-x] [-y]
    filename[.ext]
```

filename is the name of the Embedded SQL/COBOL source file. The default extension for *filename* is ".pco." cobpre generates an output file with a ".cbl" extension.

Note The precompilers for 64-bit applications are cobpre64 and cobpre_r64. The cobpre64 is the non-reentrant precompiler, and cobpre_r64 is the reentrant version.

Some of the options are switches that activate features of the precompiler, such as generating stored procedures. By default, these features are off, you can turn turn them on by including the option on the cobpre command line. Other command qualifiers specify values for the preprocessor, for example, a password. Enter the value after the option (with or without intervening spaces).

If you enter an invalid option, the precompiler lists the options that are available.

See Appendix A, "Utility Commands Reference," for detailed descriptions of the cobpre options.

Compiling and linking the application

The following tables list the general forms of the commands for compiling and linking Embedded SQL/COBOL applications on Sybase-supported platforms running the UNIX operating system.

Table 5-2 shows commands for compiling and linking Embedded SQL/COBOL applications using nondebug libraries.

Table 5-2: Static compile-and-link commands for Embedded SQL/COBOL

Platform	Command
Solaris SPARC 32-bit and 64-bit	`cob -x program.cbl -L $SYBASE/$SYBASE_OCS/lib -lsybcobct -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lnsl -ldl -lm -lsocket -o program`
Solaris x86-64 32-bit and 64-bit	`cob -x program.cbl -L $SYBASE/$SYBASE_OCS/lib -lsybcobct -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lnsl -ldl -lm -lsocket -o program`
HP HP-UX PA-RISC 32-bit and 64 bit	`cob -x program.cbl -L $SYBASE/$SYBASE_OCS/lib -lsybcobct -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lBSD -lcl -lm -o program`
HP HP-UX Itanium 32-bit and 64-bit	`cob -x program.cbl -L $SYBASE/$SYBASE_OCS/lib -lsybct -lsybcobct -lsybtcl -lsybcs -lsybcomn -lsybintl -lsybunic -lcl -lm -ldld -o program`
IBM AIX RS/6000 32-bit and 64-bit	`cob -x program.cbl -L $SYBASE/$SYBASE_OCS/lib -lsybcobct -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lm -o program`
Linux x86 32-bit	`cob -x program.cbl -L $SYBASE/$SYBASE_OCS/lib -lsybct -lsybcobct -lsybtcl -lsybcs -lsybcomn -lsybintl -lsybunic-ldl -lnsl -lm -o program`

Table 5-3 shows commands for compiling and linking Embedded SQL/COBOL applications using debug libraries.

Table 5-3: Debug compile-and-link commands for Embedded SQL/COBOL

Platform	Command
Solaris SPARC 32-bit and 64-bit	`cob -g -x program.cbl -L $SYBASE/$SYBASE_OCS/devlib -lsybcobct -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lnsl -ldl -lm -lsocket -o program`
Solaris x86-64 32-bit and 64-bit	`cob -g -x program.cbl -L $SYBASE/$SYBASE_OCS/devlib -lsybcobct -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lnsl -ldl -lm -lsocket -o program`
HP HP-UX PA-RISC 32-bit and 64 bit	`cob -g -x program.cbl -L $SYBASE/$SYBASE_OCS/devlib -lsybcobct -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lcl -lm -o program`
HP HP-UX Itanium 32-bit and 64-bit	`cob -g -x program.cbl -L $SYBASE/$SYBASE_OCS/devlib -lsybct -lsybcobct -lsybtcl -lsybcs -lsybcomn -lsybintl -lsybunic -lcl -lm -ldld -o program`
IBM AIX RS/6000 32-bit and 64-bit	`cob -g -x program.cbl -L $SYBASE/$SYBASE_OCS/devlib -lsybcobct -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lm -o program`
Linux x86 32-bit	`cob -g -x program.cbl -L $SYBASE/$SYBASE_OCS/devlib -lsybct -lsybcobct -lsybtcl -lsybcs -lsybcomn -lsybintl -lsybunic-ldl -lnsl -lm -o program`

To build a 64-bit COBOL application, make sure that the build mode for the COBOL compiler is correctly set. For example, in ESQL/COBOL, the COBMODE environment variable must be set to 32 for a 32-bit build, and to 64 for a 64-bit build. Failure to do this may result in a build error, or produce an executable with an unexpected signature on platforms that support both 32-bit and 64-bit COBOL applications. For more information about building and linking a 64-bit application, see the sybopts.sh script in *SYBASE/$SYBASE_OCS/sample/esqlcob*.

Additional considerations

Consider the following alignment issues when compiling and linking your application.

Data alignment on a 64-bit architecture

When you are building a 64-bit application, your data structure must be aligned on an 8-byte boundary, that is, to memory addresses that are multiples of 8 bytes. Similarly, the data structure of 32-bit applications must be aligned on a 4-byte boundary.

The following example illustrates this concept by creating a 32-bit and a 64-bit ESQL/COBOL version of the SQLDA, which is a descriptor area that describes objects that are referenced in dynamic SQL. The Sybase version of the SQLDA, written in C, is given as a reference in the following example.

Example

This code snippet shows the SQLDA layout supplied by Sybase:

```
typedef struct _sqlda
{
   CS_SMALLINT sd_sqln;
   CS_SMALLINT sd_sqld;
   struct _sd_column
   {
      CS_DATAFMT sd_datafmt;
      CS_VOID *sd_sqldata;
      CS_SMALLINT sd_sqlind;
      CS_INT sd_sqllen;
      CS_VOID *sd_sqlmore;
   } sd_column[1];

} syb_sqlda;

typedef syb_sqlda SQLDA;
```

The following SQLDA structure shows the 32-bit ESQL/COBOL version of the Sybase-specific SQLDA:

```
01 OUT-DES. /* 32bit */
   09 SD-SQLN PIC S9(4) COMP.
   09 SD-SQLD PIC S9(4) COMP.
   09 SD-COLUMN OCCURS 27 TIMES. /* 27-column table*/
      19 SD-DATAFMT.
         29 SQL--NM PIC X(256).
         29 SQL--NMLEN PIC S9(9) COMP.
         29 SQL--DATATYPE PIC S9(9) COMP.
         29 SQL--FORMAT PIC S9(9) COMP.
         29 SQL--MAXLENGTH PIC S9(9) COMP.
         29 SQL--SCALE PIC S9(9) COMP.
         29 SQL--PRECISION PIC S9(9) COMP.
         29 SQL--STTUS PIC S9(9) COMP.
         29 SQL--COUNT PIC S9(9) COMP.
         29 SQL--USERTYPE PIC S9(9) COMP.
      29 SQL--LOCALE PIC S9(9) COMP.
   19 SD-SQLDATA PIC S9(9) COMP.
   19 SD-SQLIND PIC S9(4) COMP.
   19 FILLER PIC S9(4) COMP. /* Filler record to */
                             /* align SQLIND   */
   19 SD-SQLLEN PIC S9(9) COMP.
   19 SD-SQLMORE PIC S9(9) COMP.
```

In the 32-bit ESQL/COBOL version of the Sybase-specific SQLDA, the picture (PIC) clauses of relevance are:

- Elements defined as S9(4) – S9(4) is the ESQL/COBOL equivalent of smallint, which is 2 bytes in length. On its own, an element defined as S9(4) does not meet the 32-bit data alignment requirement. However, an S9(4) pair, as in the case of SD-SQLN and SD-SQLD, meets this requirement because, together, the elements occupy a memory address that is a multiple of 4 bytes.
- Elements defined as S9(9) – S9(9) is the ESQL/COBOL equivalent of an int, which is 4 bytes in length. Elements defined as S9(9) meet the 32-bit data alignment requirement.
- FILLER – a filler record 2 bytes in length is added to pad SD-SQLIND, which is an unpaired S9(4) element, and to align the entire structure on a 4-byte boundary.

The following SQLDA structure shows the 64-bit ESQL/COBOL version of the Sybase-specific SQLDA. In a 64-bit environment, the entire data structure must align on an 8-byte boundary:

```
01 OUT-DES. /* 64 bit */
   09 SD-SQLN PIC S9(4) COMP.
   09 SD-SQLD PIC S9(4) COMP.
   09 FILLER PIC S9(9) COMP. /* First filler to align */
                             /* on eight bytes */
   09 SD-COLUMN OCCURS 27 TIMES. /* 27-column table */
      19 SD-DATAFMT.
         29 SQL--NM PIC X(256).
         29 SQL--NMLEN PIC S9(9) COMP.
         29 SQL--DATATYPE PIC S9(9) COMP.
         29 SQL--FORMAT PIC S9(9) COMP.
         29 SQL--MAXLENGTH PIC S9(9) COMP.
         29 SQL--SCALE PIC S9(9) COMP.
         29 SQL--PRECISION PIC S9(9) COMP.
         29 SQL--STTUS PIC S9(9) COMP.
         29 SQL--COUNT PIC S9(9) COMP.
         29 SQL--USERTYPE PIC S9(9) COMP.
         29 FILLER PIC S9(9) COMP. /* Second filler */
   29 SQL--LOCALE PIC S9(18) COMP. /* locale is */
                                /* now eight bytes */
19 SD-SQLDATA PIC S9(18) COMP.  /* SQLDATA is */
                               /* now eight bytes */
19 SD-SQLIND PIC S9(4) COMP.
19 FILLER PIC S9(4) COMP. /* Third filler */
19 SD-SQLLEN PIC S9(9) COMP.
19 SD-SQLMORE PIC S9(18) COMP. /* SQLMORE is */
                                /* now eight bytes */
```

In the 64-bit ESQL/COBOL version of the SQLDA, the PIC clauses of relevance are:

- Elements defined as S9(4) – S9(4) is equivalent to an ESQL/COBOL smallint, which is 2 bytes in length. On its own, an element defined as S9(4) does not meet the 64-bit requirement because the 64-bit architecture requires that memory addresses be in multiples of 8. To meet the requirement, an S9(4) element must be grouped with other elements or padded using a filler. In the 64-bit version of the SQLDA above, the combined length of SD-SQLN and SD-SQLD is only 4 bytes, thus, a filler 4 bytes in length is added after SD-SQLD.
- Elements defined as S9(9) – S9(9) is equivalent to an ESQL/COBOL int, which is 4 bytes in length. A pair of S9(9) elements, such as SQL-NMELEN and SQL-DATATYPE, meets the 64-bit alignment requirement.
- Elements defined as S9(18) – S9(18) is the ESQL/COBOL equivalent of a pointer or long, which is 8 bytes in length. Elements defined as S9(18) meet the 64-bit data alignment requirement.
- FILLER – in the above example, three fillers of varying lengths are used to pad and align the data structure on an 8-byte boundary.

Note Although you can use fillers to pad and align the SQLDA data structure, do not modify the SQLDA data structure. You cannot add or delete an SQLDA element, or edit the element's current definition.

Loading stored procedures

If you use the precompiler -G flag to generate stored procedures, use isql to load the stored procedures into Adaptive Server before you execute the program. The format of the isql command to execute a generated script is:

```
isql -Uuserid -Ppassword < program.sql
```

where the -U and -P flags specify the user ID and password to log in to Adaptive Server.

See Appendix A, "Utility Commands Reference," for a description of isql.

Using Embedded SQL/COBOL sample programs

The Embedded SQL/COBOL precompiler has two sample programs, described in the following sections, that demonstrate typical Embedded SQL applications.

Note Before compiling and running the sample programs, copy the contents of *$SYBASE/$SYBASE_OCS/sample/esqlcob* into a working directory, where you can experiment with the sample programs without affecting the integrity of the original files.

Purpose of the sample programs

The sample programs demonstrate specific Embedded SQL/COBOL functionality. These programs are designed as guides for application programmers, not as Embedded SQL/COBOL training aids. Read the descriptions at the top of each source file and examine the source code before using the sample programs.

Edit the samples. Before you precompile the programs, replace the user name and password with values that are valid for your Adaptive Server. Comments in the programs show where you should make the changes. See the *README* file for complete instructions on running the sample programs.

Note These simplified programs are not intended for use in a production environment. Production-quality programs require additional code to handle errors and special cases.

Location

The sample programs are located in *$SYBASE/$SYBASE_OCS/sample/esqlcob*.

This directory includes:

- Source code for the sample programs.
- The *makefile* provided to build the samples. Use the *makefile* as a starting point for your own Server-Library applications.
- The *README* file containing instructions for building, executing, and testing the samples.

Set execute permission on the *sybopts.sh* file for the file's owner:

```
chmod u+x sybopts.sh
```

Note Before the sample programs provide results, you may need to press Enter.

example1.pco sample program

The *example1.pco* shows how to use cursors in an interactive query program. The program:

- Displays a list of book types; the user selects one type
- Displays all titles in the selected book type; prompts for a title ID
- Displays detailed information about the selected title and continues prompting for title IDs
- Exits the program when Enter is pressed at a prompt

example2.pco sample program

The *example2.pco* demonstrates updating a row through a cursor. The program:

- Displays the columns in the authors table row by row.

- Lets the user update author information in all but the au_id column. If the user presses Return for column information, that column’s data remains unchanged.
- Requires the user to confirm the update before sending the data to Adaptive Server.

APPENDIX A Utility Commands Reference

This appendix contains information about these utility program commands:

Utility	Description	Page
bcp	Bulk-copy utility, which copies a database table to or from an operating system file in a user-specified format.	87
cpre	C precompiler utility, which precompiles a C source program to produce target, listing, and isql files.	111
cobpre	COBOL precompiler utility, which precompiles a COBOL source program to produce target, listing, and isql files.	121
defncopy	Definition copy utility, which copies definitions for specified views, rules, defaults, triggers, procedures, or reports from a database to an operating system file or from an operating system file to a database.	131
isql	Interactive SQL parser, which connects to and queries an Adaptive Server or Open Server.	136

bcp

Description

Copies a database table to or from an operating system file in a user-specified format. This utility is in *$SYBASE/$SYBASE_OCS/bin*.

Syntax

bcp [[*database_name*.]*owner*.]*table_name* [*:slice_number* | partition *partition_name*] {in | out} [*datafile*]
  [-a *display_charset*]
  [-A *packet_size*]
  [-b *batch_size*]
  [-c]
  [-C]
  [-d *discardfileprefix*]
  [-e *errfile*]

```
[-E]
[-f formatfile]
[-F firstrow]
[-g id_start_value]
[-i input_file]
[-I interfaces_file]
[-J client_character_set]
[-K keytab_file]
[-L lastrow]
[-m maxerrors]
[-MLabelName LabelValue] [-labeled]
[-n]
[-N]
[-o output_file]
[-P password]
[-Q]
[-r row_terminator]
[-R remote_server_principal]
[-S server]
[-t field_terminator]
[-T text_or_image_size]
[-U username]
[-v]
[-V [security_options]]
[-W]
[-x trusted.txt_file]
[-X]
[-y alternate_home_directory]
[-Y ]
[-z language]
[-Z security_mechanism]
[--colpasswd [[[db_name.[owner].]table_name.]
            column_name [password]]]
[--keypasswd [[db_name.[owner].]key_name [password]]]
[--hide-vcc]
[--initstring "TSQL_command"]
[--maxconn maximum_connections]
[--show-fi]
[--skiprows nSkipRows]
```

Parameters

database_name

Optional if the table being copied is in your default database or in *master*. Otherwise, you must specify a database name.

owner

Optional if you or the database owner owns the table being copied. If you do not specify an owner, bcp looks first for a table of that name owned by you. Then it looks for one owned by the database owner. If another user owns the table, you must specify the owner name or the command fails.

table_name
The name of the database table to copy. The table name cannot be a Transact-SQL reserved word.

slice_number
The number of the slice of the database table to copy.

partition *partition_name*
The name of the partition in Adaptive Server. For multiple partitions, use a comma-separated list of partition names.

in | out
The direction of the copy. in indicates a copy from a file into the database table, while out indicates a copy to a file from the database table.

Note bcp raises an error and stops its operation if the number of rows to be copied in or out exceeds 2147483647.

---

datafile
The full path name of an operating system file. The path name can be from 1 – 255 characters in length. For multiple files, use a comma-separated list of file names. If you enter more than one data file and partition name, the number of files and partitions must be the same.

-a *display_charset*
Allows you to run bcp from a terminal where the character set differs from that of the machine on which bcp is running. Use -a in conjunction with -J specifies the character set translation file (*.xlt* file) required for the conversion. Use -a without -J only if the client character set is the same as the default character set.

You see this error message if any character translation files are missing, or if you enter file names incorrectly:

```
Error in attempting to determine the size of a pair of
translation tables. : ‘stat’ utility failed.
```

-A *packet_size*
Specifies the network packet size to use for this bcp session. For example, the following example sets the packet size to 4096 bytes for this bcp session:

```
bcp pubs2..titles out table_out -A 4096
```

packet_size must be between the values of the default network packet size and maximum network packet size configuration variables, and it must be a multiple of 512.

Use larger-than-default network packet sizes to improve the performance of large bulk-copy operations.

-b *batchsize*

The number of rows per batch of data copied. By default, bcp in copies *n* rows in one batch, where *n* is equal to the batch size. Batch size applies only when bulk copying in; it has no effect on bulk copying out. The smallest number bcp accepts for *batchsize* is 1.

Note Setting *batchsize* to 1 causes Adaptive Server to allocate one data page to one row copied in. This parameter applies only to fast bcp, and is useful only for locating corrupt rows of data. Use -b 1 carefully, as doing so causes a new page to be allocated for each row, which is generally a poor use of space.

-c

Performs the copy operation using the char datatype as the default. This option does not prompt for each field; it uses char as the default storage type, no prefixes, \t (tab) as the default field terminator, and \n (newline) as the default row terminator.

-C

Supports bulk copy of encrypted columns if Adaptive Server supports encrypted columns. -C enables the ciphertext option before initiating the bulk copy operation.

-d *discardfileprefix*

Logs the rejected rows into a dedicated discard file. The discard file has the same format as the host file and is created by appending the input file name to the discard file prefix supplied. You can correct the rows in this file and use the file to reload the corrected rows.

Sybase recommends that you use -d *discardfileprefix* with -e *errorfile* to help identify and diagnose problem rows in the discard file.

-e *errfile*

The full path name of an error file where bcp stores all rows that bcp was unable to transfer from the file to the database. The error messages from bcp appear on your terminal, and are also logged in the error file. bcp creates an error file only when you specify this parameter. If multiple sessions are used, the partition and file name information for the error is added to the error file.

Sybase recommends that you use -e *errorfile* with -d *discardfileprefix* to help identify and diagnose problem rows in the discard file.

-E

Explicitly specifies the value of a table's IDENTITY column.

By default, when you bulk-copy data into a table with an IDENTITY column, bcp assigns each row a temporary IDENTITY column value of 0. This is effective only when copying data into a table. bcp reads the value of the ID column from the data file, but does not send it to the server. Instead, as bcp inserts each row into the table, the server assigns the row a unique, sequential IDENTITY column value, beginning with the value 1. If you specify the -E flag when copying data into a table, bcp reads the value from the data file and sends it to the server, which inserts the value into the table. If the number of inserted rows exceeds the maximum possible IDENTITY column value, Adaptive Server returns an error.

By default, when you bulk copy data from a table with an IDENTITY column, bcp excludes all information about the column from the output file. If you specify the -E flag, bcp copies the existing IDENTITY column values into the output file.

The -E parameter has no effect when you are bulk copying data out. Adaptive Server copies the ID column to the data file, unless you use the -N parameter.

You cannot use the -E and -g flags together.

-f *formatfile*

The full path name of a file with stored responses from a previous use of bcp on the same table. After you answer bcp's format questions, it prompts you to save your answers in a format file. Creation of the format file is optional. The default file name is *bcp.fmt*. The bcp program can refer to a format file when copying data, so that you need not interactively duplicate your previous format responses. Use this parameter only if you previously created a format file that you want to use now for a copy in or out. If you do not use this option, you must interactively enter format information.

-F *firstrow*
: The number of the first row to copy (the default is the first row). If you use multiple files, this option applies to each file.

  Do not use this parameter when performing heavy-duty, multiprocess copying, as it causes bcp to generally spend more effort to run, and does not provide you with a faster process. Instead, use -F for single-process, ad hoc copying.

Note You cannot use -F with --skiprows.

-g *id_start_value*
: Specifies the value of the IDENTITY column to use as a starting point for copying data in.

  You cannot use the -g and -E flags together.

-i *input_file*
: Specifies the name of the input file. Standard input (stdin) is the default.

-I *interfaces_file*
: Specifies the name and location of the interfaces file to search when connecting to Adaptive Server. If you do not specify -I, bcp looks for the interfaces file, *interfaces*, in the Sybase release directory.

-J *client_character_set*
: Specifies the character set to use on the client. bcp uses a filter to convert input between *client_charset* and the Adaptive Server character set.

  -J *client_character_set* requests that Adaptive Server convert to and from *client_character_set*, the character set used on the client.

  -J with no argument disables character set conversion. No conversion takes place. Use this if the client and server use the same character set.

  Omitting -J sets the character set to a default for the platform, which may not necessarily be the character set that the client is using. For more information about character sets and associated flags, see the *Adaptive Server Enterprise System Administration Guide*.

-K *keytab_file*
: (Used only with DCE security). Specifies a DCE keytab file that contains the security key for the user name specified with -U option. Create keytab with the DCE dcecp utility. See your DCE documentation.

  If the -K option is not supplied, the bcp user must be logged in to DCE with the same user name as specified with the -U option.

-L *lastrow*

The number of the last row to copy from an input file (the default is the last row). If you use multiple files, this option applies to each file.

-m *maxerrors*

The maximum number of errors permitted before bcp aborts the copy. bcp discards each row that it cannot insert (due to a data conversion error, or an attempt to insert a null value into a column that does not allow them), each rejected row as one error. If you do not include this option, bcp uses a default value of 10.

If you use multiple partitions, the same number of *maxerrors* is used for every file.

-M *LabelName LabelValue*

(secure SQL Server only) enables multilevel users to set the session labels for the bulk-copy. Valid values for *LabelName* are:

- curread (current read level) is the initial level of data that you can read during this session. curread must dominate curwrite.
- curwrite (current write level) is the initial sensitivity level that is applied to any data that you write during this session.
- maxread (maximum read level) is the maximum level at which you can read data. This is the upper bound to which you as a multilevel user can set curread during the session. maxread must dominate maxwrite.
- maxwrite (maximum write level) is the maximum level at which you can write data. This is the upper bound to which you as a multilevel user can set curwrite during a session. maxwrite must dominate minwrite and curwrite.
- minwrite (minimum write level) is the minimum level at which you can write data. This is the lower bound to which you as a multilevel user can set curwrite during a session. minwrite must be dominated by maxwrite and curwrite.

LabelValue is the actual value of the label, expressed in the human-readable format used on your system (for example, “Company Confidential Personnel”).

-labeled

(secure SQL Server only) indicates that the data you are importing already has labels in the first field of every record.

For exporting data -labeled indicates that you want the sensitivity label of every row to be copied out as the first field.

-n
Performs the copy operation using native (operating system) formats. Specifying the -n parameter means bcp does not prompt for each field. Files in native data format are not human-readable.

Warning! Do not use bcp in native format for data recovery, salvage, or to resolve an emergency situation. Do not use bcp in native format to transport data between different hardware platforms, different operating systems, or different major releases of Adaptive Server. Do not use field terminators (-t) or row terminators (-r) with bcp in native format. Results are unpredictable and data may get corrupted. Using bcp in native format can create flat files that cannot be reloaded into Adaptive Server, and it may be impossible to recover the data. If you cannot re-run bcp in character format (for example, a table was truncated or dropped, hardware damage occurred, a database table was dropped, and so on), the data is unrecoverable.

-N
Skips the IDENTITY column. Use this option when copying data in if your host data file does not include a placeholder for the IDENTITY column values, or when copying data out and you do not want to include the IDENTITY column information in the host file.

You cannot use both -N and -E options when copying in data.

-o *output_file*
Specifies the name of the output file. Standard output (stdout) is the default.

-P *password*
Specifies an Adaptive Server password. If you do not specify -P *password*, bcp prompts for a password. You can leave out the -P flag if your password is NULL.

-Q
Provides backward compatibility with bcp for copying operations involving nullable columns.

-r *row_terminator*
Specifies the row terminator.

-R *remote_server_principal*
Specifies the principal name for the server. By default, a server's principal name matches the server's network name (which is specified with the -S option or the DSQUERY environment variable). Use the -R option when the server's principal name and network name are not the same.

-S *server*
Specifies the name of the Adaptive Server to connect to. If you specify -S with no argument, bcp uses the server specified by your DSQUERY environment variable.

-t *field_terminator*
Specifies the default field terminator.

-T *text_or_image_size*
Allows you to specify, in bytes, the maximum length of text or image data that Adaptive Server sends. The default is 32K. If a text or image field is larger than the value of -T or the default, bcp does not send the overflow.

-U *username*
Specifies an Adaptive Server login name. If you do not specify *username*, bcp uses the current user's operating system login name.

-v
Displays the current version of bcp and a copyright message and returns to the operating system.

SDK binaries like bcp have the same names in both the 32-bit and 64-bit products. Installing Adaptive Server, the SDK, or Open Server 64-bit products with other Sybase 32-bit products overwrites the 32-bit binaries. Starting with Adaptive Server 15.0.2 and SDK/Open Server 15.0 ESD #9, the 64-bit binaries have been replaced with 32-bit binaries on all 64-bit UNIX platforms. Since 32-bit binaries are included in the 64-bit EBF, the -v option of bcp is no longer a valid way to check the EBF number for 64-bit products. Instead, use the UNIX strings and grep commands to confirm the EBF numbers for both Open Client and Open Server.

For example, to find the string containing the EBF number in the *libsybct64.a* library, enter:

```
strings -a libsybct64.a | grep Sybase
```

This returns a string similar to:

```
Sybase Client-Library/15.5/P/DRV.15.5.0/SPARC/Solaris
8/BUILD1550-001/64bit/OPT/Mon Aug 10 23:04:17 2009
```

To find the string containing the EBF number in the *libsybsrv64.a* library, enter the following:

```
strings -a libsybsrv64.a | grep Sybase
```

This returns a string like the following:

```
Sybase Server-Library/15.5/P/DRV.15.5.0/SPARC/Solaris
```

```
8/BUILD1550-001/64bit/OPT/Mon Aug 10 23:06:27 2009
```

-V *security_options*

Specifies network-based user authentication. With this option, the user must log in to the network's security system before running the utility. In this case, users must supply their network user name with the -U option; any password supplied with the -P option is ignored.

-V can be followed by a *security_options* string that enables additional security services:

- c – enable data confidentiality service.
- d – enable credential delegation and forward the client credentials to the gateway application.
- i – enable data integrity service.
- m – enable mutual authentication for connection establishment.
- o – enable data origin stamping service.
- q – enable out-of-sequence detection.
- r – enable data replay detection.

-W

Specifies that if the server to which bcp is attempting to connect supports neither normal password encryption nor extended password encryption, plain text password retries are disabled. If this option is used, the CS_SEC_NON_ENCRYPTION_RETRY connection property is set to CS_FALSE, and plain text (unencrypted) passwords are not used in retrying the connection.

-x *trusted.txt_file*

Specifies an alternate *trusted.txt* file

-X

Specifies that, in this connection to the server, the application initiates the login with client-side password encryption. bcp (the client) specifies to the server that password encryption is desired. The server sends back an encryption key, which bcp uses to encrypt your password, and the server uses the key to authenticate your password when it arrives.

This option can result in normal or extended password encryption, depending on connection property settings at the server. If CS_SEC_ENCRYPTION is set to CS_TRUE, normal password encryption is used. If CS_SEC_EXTENDED_ENCRYPTION is set to CS_TRUE, extended password encryption is used. If both CS_SEC_ENCRYPTION and CS_SEC_EXTENDED_ENCRYPTION are set to CS_TRUE, extended password encryption is used as the first preference.

If bcp fails, the system creates a core file that contains your password. If you did not use the encryption option, the password appears in plain text in the file. If you used the encryption option, your password is not readable.

-y *alternate_home_directory*

Sets an alternate Sybase home directory.

-Y

Specifies that the character set conversion is disabled in the server, and is performed by bcp on the client side when using bcp out.

Note All character set conversion is done in the server during bcp out.

-z *language*

The official name of an alternate language that the server uses to display bcp prompts and messages. Without the -z flag, bcp uses the server's default language.

You can add languages to an Adaptive Server during installation or afterwards, using either the langinst utility or the sp_addlanguage stored procedure.

The following error message appears if an incorrect or unrecognized language is named with the -z parameter:

```
Unrecognized localization object. Using default value 'us_english'.
Starting copy ...
=> warning.
```

-Z *security_mechanism*
Specifies the name of a security mechanism to use on the connection.

Security mechanism names are defined in the *libtcl.cfg* configuration file, which is located in *$SYBASE/$SYBASE_OCS/config*. If no *security_mechanism* name is supplied, the default mechanism is used.

Note The CS_LIBTCL_CFG property specifies the name and path to an alternative *libtcl.cfg* file. For details about this property, see the *Open Client and Open Server Client Libraries Reference Manual*.

For more information about security mechanism names, see the description of the *libtcl.cfg* file in the *Open Client and Open Server Configuration Guide for UNIX*.

--colpasswd *column_name password*
Sets passwords for encrypted columns by sending "set encryption passwd *password* for column *column_name*" to Adaptive Server. This does not automatically apply passwords to other encrypted columns, even if the second column is encrypted with the same key. Supply the password a second time to access the second column.

--hide-vcc
Instructs bcp not to copy virtual computed columns (VCC) either to or from a data file. When you use this parameter in bcp OUT, the datafile does not contain data for VCC; in bcp IN, the data file may not contain data for a VCC.

If you use this option, Adaptive Server does not calculate or send virtual computed column data.

--initstring *"TSQL_command"*
Sends Transact-SQL commands to Adaptive Server before data is transferred.

Result sets issued by the initialization string are silently ignored, unless an error occurs. If Adaptive Server returns an error, bcp stops before data is transferred, and displays an error message.

--keypasswd *key_name password*
Sets passwords for all columns accessed by a key by sending "set encryption passwd *password* for key *key_name*" to Adaptive Server.

--maxconn *maximum_connections*

The maximum number of parallel connections permitted for each bulk copy operation. You must use bcp_r, the threaded version of the bcp utility, to copy multiple files in parallel. For example, the following example sets the maximum number of parallel connection permitted for each operation to 2:

```
bcp_r --maxconn 2
```

If you do not include this parameter, bcp uses a default value of 10.

--show-fi

Instructs bcp to copy functional indexes, while using either bcp IN or bcp OUT. If you do not specify this parameter, Adaptive Server generates the value for the functional index.

--skiprows *nSkipRows*

Instructs bcp to skip a specified number of rows before starting to copy from an input file. The valid range for --skiprows is between 0 and the actual number of rows in the input file. If you provide an invalid value, you see an error message.

Note You cannot use --skiprows with the -F option.

Examples

Example 1 The -c option copies data out of the publishers table in character format (using char for all fields). The -t field_terminator option ends each field with a comma, and the -r row_terminator option ends each line with a Return. bcp prompts only for a password. The first backslash before the final "r" escapes the second so that only one backslash prints:

```
bcp pubs2..publishers out pub_out -c -t , -r \\r
```

Example 2 The -C parameter copies data out of the publishers table (with encrypted columns) in cipher-text format instead of plain text. Press Return to accept the defaults specified by the prompts. The same prompts appear when copying data into the publishers table.

```
bcp pubs2..publishers out pub_out -C
Password:
Enter the file storage type of field col1 [int]:
Enter prefix length of field col1 [0]:
Enter field terminator [none]:
Enter the file storage type of field col2 [char]:
Enter prefix length of field col2 [0]:
Enter length of field col2 [10]:
Enter field terminator [none]:
Enter the file storage type of field col3 [char]:
```

```
Enter prefix length of field col3 [1]:
Enter field terminator [none]:
```

Example 3 Copies data from the publishers table to a file named *pub_out* for later reloading into Adaptive Server. Press Return to accept the defaults that the prompts specify. The same prompts appear when copying data into the publishers table.

```
 bcp pubs2..publishers out pub_out
 Password:

 Enter the file storage type of field pub_id [char]:
 Enter prefix length of field pub_id [0]:
 Enter length of field pub_id [4]:
 Enter field terminator [none]:
 Enter the file storage type of field pub_name [char]:
 Enter prefix length of field pub_name [1]:
 Enter length of field pub_name [40]:
 Enter field terminator [none]:
 Enter the file storage type of field city [char]:
 Enter prefix length of field city [1]:
 Enter length of field city [20]:
 Enter field terminator [none]:

 Enter the file storage type of field state [char]:
 Enter prefix length of field state [1]:
 Enter length of field state [2]:
 Enter field terminator [none]:
```

You are then asked:

```
Do you want to save this format information in a
    file? [Y-n] y
Host filename [bcp.fmt]: pub_form
Starting copy...
3 rows copied.
 Clock time (ms.): total = 1 Avg = 0 (3000.00 rows per
sec.)
```

Example 4 Copies data out of partition p1 of table t1 to the *mypart.dat* file in the current directory:

```
bcp t1 partition p1 out mypart.dat
```

Example 5 Copies data back into Adaptive Server using the saved format file, *pub_form*:

```
bcp pubs2..publishers in pub_out -f pub_form
```

Example 6 Copies a data file created with a character set used on a VT200 terminal into the pubs2..publishers table. The -z flag displays bcp messages in French:

```
bcp pubs2..publishers in vt200_data -J iso_1 -z french
```

Example 7 Copies files data.first, data.last and data.other into partitions *p1*, *p2*, and *p3*, respectively:

```
bcp t1 partition p1, p2, p3 in data.first, data.last,
data.other
```

Example 8 Copies the *mypart.dat* file from the current directory, into table t1 of partition p1:

```
bcp t1 partition p1 in mypart.dat
```

Example 9 Copies partitions p1, p2 and p3 to files *a*, *b*, and *c*, respectively, in the *\work2\data* directory:

```
bcp t1 partition p1, p2, p3 out \work2\data\a,
\work2\data\b, \work2\data\c
```

Example 10 Copies files data.first, data.last and data.other into partitions *p1*, *p2*, and *p3*, respectively:

```
bcp t1 partition p1, p2, p3 in data.first, data.last,
data.other
```

Example 11 Disables replication when *titles.txt* data is transferred into the pubs2 titles table:

```
bcp pubs2..titles in titles.txt -- initstring “set
replication off”
```

Note Because the set replication off command in this example is limited to the current session in Adaptive Server, there is no need to explicitly reset the configuration option when bcp is finished.

Example 12 Sets the password to pwd1 for the encrypted column col1:

```
bcp mydb..mytable out myfile –U uuu –P ppp --colpasswd
db..tbl.col1 pwd1
```

Example 13 Sets a prompt to enter the password for encrypted column col1:

```
bcp mydb..mytable out myfile –U uuu –P ppp --colpasswd
db..tbl.col1
Enter column db..tbl.col1’s password: ***?
```

Example 14 Reads the password for encrypted column col1 from an external OS file named “passwordfile”:

```
bcp mydb..mytable out myfile –U uuu –P ppp --colpasswd
db..tbl.col1 < passwordfile
```

Example 15 Sets password pwd1 for encryption key key1:

```
bcp mydb..mytable in myfile –U uuu -p ppp --keypasswd
db..key1 pwd1
```

Example 16 Creates the discard file *reject_titlesfile.txt*:

```
bcp pubs2..titles in titlesfile.txt -d reject_
```

Example 17 For MIT Kerberos, requests credential delegation and forwards the client credentials to MY_GATEWAY:

```
bcp -Vd -SMY_GATEWAY
```

Example 18 bcp ignores the first two rows of the input file *titles.txt*, and starts to copy from the third row:

```
bcp pubs2..titles in titles.txt -U username -P password
--skiprows 2
```

Example 19 Sets an alternate Sybase home directory:

```
bcp tempdb..T1 out T1.out -y/work/NewSybase -Uuser1
-Psecret -SMYSERVER
```

Usage

- bcp_r is a threaded version of bcp. You must use bcp_r if a security service, such as Kerberos, or a directory service, such as LDAP, is used.
- You cannot use named pipes to copy files in or out.
- Using --hide-vcc improves performance, as Adaptive Server does not transfer and calculate data from virtual computed columns.
- Although you can use any Transact-SQL command with --initstring as an initialization string for bcp, you must reset possible permanent changes to the server configuration after running bcp. You can, for example, reset changes in a separate isql session.
- *slice_number* is included for backward compatibility with Adaptive Server 12.5.x and earlier, and can be used only with round-robin-partitioned tables.
- You can specify either *slice_number* or partition *partition_name*, not both.
- If you do not specify *partition_name*, bcp copies to the entire table.

- You can specify multiple partitions and data files. Separate each partition name or data file with commas.
- bcp provides a convenient and high-speed method for transferring data between a database table or view and an operating system file. bcp can read or write files in a wide variety of formats. When copying in from a file, bcp inserts data into an existing database table; when copying out to a file, bcp overwrites any previous contents of the file.
- Upon completion, bcp informs you of the number of rows of data successfully copied, the total time the copy took, the average amount of time in milliseconds that it took to copy one row, and the number of rows copied per second.
- bcp does not insert any row that contains an entry exceeding the character length of the corresponding target table column. For example, bcp does not insert a row with a field of 300 bytes into a table with a character-column length of 256 bytes. Instead, bcp reports a conversion error and skips the row. bcp does not insert truncated data into the table. The conversion error is as follows:

  ```
  cs_convert: cslib user api layer: common library
  error: The result is truncated because the
  conversion/operation resulted in overflow
  ```

  To keep track of data that violates length requirements, run bcp with the -e log-file name option. bcp records the row and the column number of the rejected data, the error message, and the data in the log file you specify.

  To restrict the functionality of bcp to that of a previous version, set the CS_BEHAVIOR property in the [bcp] section of the *ocs.cfg* file:

  ```
  [bcp]

  CS_BEHAVIOR = CS_BEHAVIOR_100
  ```

  If CS_BEHAVIOR is not set to CS_BEHAVIOR_100, you can use functionality for bcp 11.1 and later.
- If bcp is invoked and no value is supplied for the -c, -f, or -n parameters, a bcp prompt requests the file storage type. The file storage type can be any valid Adaptive Server datatype. Storage types for the bigdatetime and bigtime Adaptive Server datatypes are specified as:

Storage type	Table datatype
A	bigdatetime
B	bigtime

- You can specify these datatypes for a bcp format file using the bigdatetime or bigtime datatypes.

Table A-1: Host file datatype storage formats

Storage format	Adaptive Server datatype
SYBBIGDATETIME	bigdatetime
SYBBIGTIME	bigtime

Using the -d option

- Specifying the -d option applies only when bulk copying in; it is silently ignored when used in bulk copying out.
- If you use multiple input files, one discard file is created for every input file that has an erroneous row. If there are no rejected rows, no discard file is created.
- If bcp reaches the maximum errors allowed and stops the operation, all the rows, from the beginning of the batch until the failed row are logged.
- If you use the -d option, the batch size is automatically adjusted. You see a warning message if you:
  - Specify -b *batchsize*, but the batch or row size is too big to hold all the rows of the batch in memory or
  - Do not specify -b *batchsize*.

Copying tables with indexes or triggers

- bcp is optimized to load data into tables that do not have indexes or triggers associated with them. It loads data into tables without indexes or triggers at the fastest possible speed, with a minimum of logging. Page allocations are logged, but row insertions are not.

  When you copy data into a table that has one or more indexes or triggers, a slower version of bcp is automatically used, which logs row inserts. This includes indexes that are implicitly created using the unique integrity constraint of a create table command. However, bcp does not enforce the other integrity constraints defined for a table.

Because the fast version of bcp inserts data without logging it, the system administrator or database owner must first set sp_dboption DBNAME,"select into/bulkcopy",true. If the option is not true, and you try to copy data into a table that has no indexes or triggers, Adaptive Server generates an error message. You need not set this option to copy data out to a file or into a table that contains indexes or triggers.

Note Because bcp logs inserts into a table that has indexes or triggers, the log can grow very large. You can truncate the log with dump transaction to truncate the log after the bulk copy completes, and after you have backed up your database with dump database.

- While the select into/bulkcopy option is on, you cannot dump the transaction log. Issuing dump transaction produces an error message instructing you to use dump database instead.

Warning! Ensure that you dump your database before you turn off the select into/bulkcopy flag. If you have inserted unlogged data into your database, and you then perform a dump transaction before performing a dump database, you cannot recover your data.

- Unlogged bcp runs slowly while a dump database is taking place.
- Table A-2 shows which version bcp uses when copying in, the necessary settings for the select into/bulkcopy option, and whether the transaction log is kept and can be dumped.

Table A-2: Comparing fast and slow bcp

	select into/ bulkcopy on	**select into/ bulkcopy off**
Fast bcp (no indexes or triggers on target table)	Yes dump transaction prohibited	No Adaptive Server forces slow bcp
Slow bcp (one or more indexes or triggers)	Yes dump transaction prohibited	Yes dump transaction OK

- By default, the select into/bulkcopy option is off in newly created databases. To change the default, turn the option on in the model database.

Note The performance penalty for copying data into a table that has indexes or triggers can be severe. If you are copying in a large number of rows, it may be faster to first use drop index (or alter table for indexes...) and drop trigger to drop all the indexes and triggers; set the database option; copy the data into the table; re-create the indexes and triggers; and then dump the database. However, you must allocate extra disk space for the construction of indexes and triggers—about 2.2 times the amount of space needed for the data.

Responding to *bcp* prompts

When you copy data in or out using the -n (native format) or -c (character format) option, bcp prompts only for your password, unless you supplied it with the -P option. If you do not supply either the -n, -c or -f *formatfile* option, bcp prompts you for information for each field in the table.

- Each prompt displays a default value, in brackets, which you can accept by pressing Return. The prompts include:
  - The file storage type, which can be character or any valid Adaptive Server datatype
  - The prefix length, which is an integer indicating the length, in bytes, of the data that follows
  - The storage length of the data in the file for non-NULL fields
  - The field terminator, which can be any character string
  - Scale and precision for numeric and decimal datatypes

  The row terminator is the field terminator of the last field in the table or file.

- The bracketed defaults represent reasonable values for the datatypes of the field in question. For the most efficient use of space when copying out to a file:
  - Use the default prompts
  - Copy all data in their table datatypes
  - Use prefixes as indicated
  - Do not use terminators
  - Accept the default lengths

  Table A-3 shows the defaults and possible alternate responses:

Table A-3: bcp prompts, their defaults and responses

Prompt	Default provided	Possible responses
File storage type	Use database storage type for most fields except: char for varchar binary for varbinary	char to create or read a human-readable file; any Adaptive Server datatype where implicit conversion is supported
Prefix length	• 0 for fields defined with datatype (not storage type) (char and all fixed-length datatype) • 1 for most other datatypes • 2 for binary and varbinary saved as char • 4 for text and image	0 if no prefix is desired; defaults are recommended in all other cases
Storage length	For char and varchar, use defined length. For binary and varbinary saved as char, use default. For all other datatypes, use maximum length needed to avoid truncation or data overflow.	Default values, or greater, are recommended
Field or row terminator	None.	Up to 30 characters, or one of: • \t tab • \n newline • \r carriage return • \0 null terminator • \ backslash

- bcp can copy data out to a file either as its native (database) datatype, or as any datatype for which implicit conversion is supported. bcp copies user-defined datatypes as their base datatype or as any datatype for which implicit conversion is supported. See dbconvert in the *Open Client DB-*

Library/C Reference Manual.

Note Be careful when you copy data from different versions of Adaptive Server, because not all versions support the same datatypes.

- A prefix length is a 1-byte, 2-byte, or 4-byte integer that represents the length of each data value in bytes. It immediately precedes the data value in the host file.
- Be sure that fields defined in the database as char, nchar, and binary are always padded with spaces (null bytes for binary) to the full length defined in the database. timestamp data is treated as binary(8).

  If data in varchar and varbinary fields is longer than the length you specify for copy out, bcp silently truncates the data in the file at the specified length.
- A field terminator string can be up to 30 characters long. The most common terminators are a tab (entered as "\t" and used for all columns except the last one), and a newline (entered as "\n" and used for the last field in a row). Other terminators are: "\0" (the null terminator), "\" (backslash), and "\r" (Return). When choosing a terminator, be sure that its pattern does not appear in any of your character data. For example, if you use tab terminators with a string that contains a tab, bcp cannot identify which tab represents the end of the string. bcp always looks for the first possible terminator, in this case, it will find the wrong one.

  When a terminator or prefix is present, it affects the actual length of data transferred. If the length of an entry being copied out to a file is smaller than the storage length, it is followed immediately by the terminator, or the prefix for the next field. The entry is not padded to the full storage length (char, nchar, and binary data is returned from Adaptive Server already padded to the full length).

  When copying in from a file, data is transferred until either the number of bytes indicated in the "Length" prompt has been copied, or the terminator is encountered. Once a number of bytes equal to the specified length has been transferred, the rest of the data is flushed until the terminator is encountered. When no terminator is used, the table storage length is strictly observed.

- Table A-4 and Table A-5 show the interaction of prefix lengths, terminators, and field length on the information in the file. "P" indicates the prefix in the stored table; "T" indicates the terminator; and dashes, "--", show appended spaces. "..." indicates that the pattern repeats for each field. The field length is 8 for each column, and "string" represents the 6-character field each time.

Table A-4: Adaptive Server char data

	Prefix length 0	**Prefix length 1, 2 or 4**
No terminator	string--string--	Pstring--Pstring--
Terminator	string--Tstring--T	Pstring--TPstring--T

Table A-5: Other datatypes converted to char storage

	Prefix length 0	**Prefix length 1, 2 or 4**
No terminator	string--string--	PstringPstring
Terminator	stringTstringT	PstringTPstringT

- The file storage type and length of a column do not have to be the same as the type and length of the column in the database table. However, if types and formats copied in are incompatible with the structure of the database table, the copy fails.
- File storage length generally indicates the maximum amount of data to be transferred for the column, excluding terminators and prefixes.
- When copying data into a table, bcp observes any defaults defined for columns and user-defined datatypes. However, bcp ignores rules to load data at the fastest possible speed.
- Because bcp considers any data column that can contain null values to be variable length, use either a length prefix or terminator to denote the length of each data row.
- Data written to a host file in its native format preserves all of its precision. datetime and float values preserve all of their precision even when they are converted to character format. Adaptive Server stores money values to a precision of one ten-thousandth of a monetary unit. However, when money values are converted to character format, their character format values are recorded only to the nearest two places.

- Before copying data in character format from a file into a database table, check the datatype entry rules in the "Datatypes" chapter of the *Adaptive Server Enterprise Reference Manual*. Character data that is copied into the database with bcp must conform to those rules. Dates in the undelimited *(yy)yymmdd* format may result in overflow errors if the year is not specified first.
- When you send host data files to sites that use terminals different from your own, inform them of the *datafile_charset* that you used to create the files.

Messages

```
Error in attempting to load a view of translation tables.
```

The character translation file specified with the -q parameter is missing, or you mistyped the name.

cpre

Description

cpre precompiles a C source program to produce target, listing, and isql files. This utility is available in the *$SYBASE/$SYBASE_OCS/bin* directory.

Note The precompilers 64-bit applications are cpre64 and cpre_r64. The cpre64 precompiler is the non-reentrant precompiler, and the cpre_r64 is the reentrant version. You can use these precompilers on any 64-bit platform supported for Open Client and Open Server

Syntax

cpre
  [-C*compiler*]
  [-D*database*_name]
  [-F*fips_level*]
  [-G[*isql_file_name*]]
  [-H]
  [-I*include_path_name*]...
  [-J*charset_locale_name*]
  [-K*syntax_level*]
  [-L[*listing_file_name*]]
  [-N*interface_file_name*]
  [-O*target_file_name*]
  [-P*password*]
  [-S*server_name*]
  [-T*tag_id*]
  [-U*user_id*]

```
 [-V*version_number*]
 [-Z*language_locale_name*]
[-a] [-b] [-c] [-d] [-e] [-f] [-h] [-l] [-m] [-p] [-r] [-s] [-u] [-v] [-w] [-x] [-y]
filename[.ext]
```

Note You can use either a slash (/) or a dash (-) to specify options; for example, cpre -l and cpre /l are equivalent.

Parameters

-C *compiler*
Specifies the target host language compiler values, such as "ANSI_C", which is an ANSI C compiler.

-D *database_name*
Specifies the name of the database to parse against. Use this option to check SQL semantics at precompile time. If you also specify -G, use *database* command is added to the beginning of the *filename.sql* file. If you do not use this option, the precompiler uses the default database on the Adaptive Server.

-F *fips_level*
Checks for the specified conformance level. Currently, the precompiler can check for SQL89 or SQL92E.

-G *isql_file_name*
Generates stored procedures for appropriate SQL statements and saves them to a file for input to the database through isql. *isql_file_name* is optional. If you have multiple input files, you may use -G, but you cannot specify an argument.

If you have multiple input files or do not specify the argument, the default target file names are the input file names with the extension .isql either appended, or replacing any existing input file name extension.

Also, see option -T tag_id to specify tag IDs for stored procedures.

If you do not use the -G option, no stored procedures are generated.

-H
Generates code with failover capability.

-I *include_path_name*
Specifies a directory complete with the path name, where Embedded SQL searches for *include* files. You can specify this option any number of times. Embedded SQL searches the directories in command line order. If you do not use this option, the default is the */include* directory of the Sybase release directory and the current working directory.

-J *charset_locale_name*

Specifies the character set of the source file that is being precompiled. The option's value must be a locale name that corresponds to an entry in the locales file. If you do not specify -J, the precompiler interprets the source file as being in the precompiler's default character set.

To determine which character set to use as the default, the precompiler looks for a locale name. CS-Library searches for the information in this order:

1. LC_ALL
2. LANG

   If LC_ALL is defined, CS-Library uses its value as the locale name. If LC_ALL is not defined but LANG is defined, CS-Library uses its value as the locale name. If none of these locale values are defined, CS-Library uses a locale name of "default."

The precompiler looks up the locale name in the *locales* file and uses the character set associated with the locale name as the default character set.

-K *syntax_level*

Specifies the level of syntax checking to perform:

- none (default)
- syntax
- semantic

If you use either SYNTAX or SEMANTIC, you must also specify the -U, -P, -S, and -D options so Embedded SQL can connect to your Adaptive Server.

If you do not use this option, the precompiler does not connect to a server or perform SQL syntax checking of the input file beyond what is required to generate the target file.

-L *listing_file_name*

Generates one or more listing files. *listing_file_name*, which is an optional argument, is a version of the input file that numbers each line, and includes any applicable error messages. If you have multiple input files, you may use -L, but you cannot specify an argument.

If you have multiple input files or do not specify the argument, the default listing file names are the input file names with the extension *.lis* either appended, or replacing any existing input file name extension.

If you do not use this option, no listing file is generated.

-N *interface_file_name*
Specifies the interface file name, *interfaces,* to the precompiler.

-O *target_file_name*
Specifies the target or output file name. If you have multiple input files, you cannot use this option; default target names are assigned—the input file name with the extension *.cbl* either appended, or replacing any existing input file name extension.

-P *password* (used with option -Uuser_id)
Specifies an Adaptive Server password for SQL syntax checking at precompile time. Using -P without an argument, or with the keyword NULL as an argument, specifies a null (““) password. If you use option -Uuser_id without using -P, the precompiler prompts you to enter a password. Must be used with the -G flag.

-S *server_name*
Specifies the name of the Adaptive Server for SQL syntax checking at precompile time. If you do not use this option, the default Adaptive Server name is taken from the DSQUERY environment variable. If DSQUERY is not set, SYBASE is used as the name of the server.

-T *tag_id* (used with option -G)
Specifies a tag ID (up to 3 characters) to append to the end of the generated stored procedure group name.

For example, if you enter -T dbg as part of your command, generated stored procedures are assigned the name of the input file with the tag ID *dbg* appended: *program_dbg;1*, *program_dbg;2*, and so on.

Programmers can use tag IDs to test changes to an existing application without destroying the existing generated stored procedures, which may be in use.

If you do not use this option, no tag ID is added to the stored procedure name.

-U *user_id*
Specifies the Adaptive Server user ID. This option allows you to check SQL syntax at precompile time. It causes the precompiler to pass SQL statements to the server for parsing only. If the server detects syntax errors, the errors are reported and no code is generated. If you are not using -Ppassword, this option prompts you to enter a password.

Also see -K, -P, -S, and -D options.

-V *version_number*

Specifies the Client-Library version number. For COBOL, the version number must match one of the values from *cobpub.cbl*. If you do not use this option, the default is the most recent version of Client-Library available with the precompiler (CS_VERSION_155 for Open Client and Open Server version 15.5).

-Z *language_locale_name*

Specifies the language and character set that the precompiler uses for messages. If you do not specify -Z, the precompiler uses its default language and character set for messages.

To determine which language and character set to use as its default for messages, the precompiler performs the following, in order:

1. Looks for a locale name. CS-Library searches for the information first in LC_ALL, then in LANG. If neither of these locale values are defined, CS-Library uses a locale name of “default.”
2. Looks up the locale name in the *locales.dat* file available in *$SYBASE/locales* directory to determine which language and character set are associated with it.
3. Loads localized messages and character set information appropriate to the language and character set determined in step 2.

@options_file

Can be used to specify a file containing any of the above command-line arguments. The precompiler reads the arguments contained in this file in addition to any arguments already specified. If the file specified with *@options_file* contains names of the files to precompile, place the argument at the end of the command line.

-a

Allows cursors to remain open across transactions. If you do not use this option, cursors behave as though set close on endtran on were in effect. This behavior is ANSI-compatible. See the *Adaptive Server Enterprise Reference Manual* for information about cursors and transactions.

-b

Disables rebinding of host variable addresses typically used in fetch statements. If you do not use this option, a rebind occurs on every fetch statement unless you specify otherwise in your Embedded SQL/C program.

The -b option differs in the 11.1 and 10.x versions of the Embedded SQL precompilers:

- For the 11.1 and later versions of cpre, the norebind attribute applies to all fetch statements of a cursor for which the declaration was precompiled with the /b option.
- For the 10.0 and earlier versions of cpre, the norebind attribute applies to all fetch statements in each Embedded SQL source file precompiled with /b, regardless of where the cursors were declared.

-c

Turns on the debugging feature of Client-Library by generating calls to ct_debug.

This option is useful during application development but should be turned off for final application delivery. The application must be linked and run with the libraries located in the *$SYBASE/$SYBASE_OCS/devlib* directory of the Sybase release directory.

-d

Turns off delimited identifiers (identifiers delimited by double quotes) and allows quoted strings in SQL statements to be treated as character literals.

-e

When processing an exec sql connect statement, directs Client-Library to use the external configuration file to configure the connection. Also see the /x option and CS_CONFIG_BY_SERVERNAME property in the *Open Client Client-Library/C Reference Manual*.

Without this option, the precompiler generates Client-Library function calls to configure the connection. See the *Open Client Client-Library/C Reference Manual* for information about the external configuration file

-f

Turns on the FIPS flagger for ANSI FIPS compliance checking.

-h

Generates thread-safe code.

-l

Turns off generation of #line directives.

-m

Runs the application in Sybase auto-commit mode, which means that transactions are not chained. Explicit begin and end transactions are required, or every statement is immediately committed. If you do not specify this option, the application runs in ANSI-style chained transaction mode.

-p

Generates a separate command handle for each SQL statement in the module that has input host variables, and enables sticky binds on each command handle. This option improves performance of repeatedly executed commands with input parameters at the cost of increased storage space usage and longer first executions of each such command.

Applications that rely on inserting empty strings instead of NULL strings when the host string variable is empty do not work if the -p option is turned on. The persistent bind implementation prevents Embedded SQL from circumventing Client-Library protocol (which inserts NULL strings).

-r

Disables repeatable reads. If you do not use this option, a set transaction isolation level 3 statement, which executes during connect statements, is generated. The default isolation level is 1.

-s

Includes static function declarations.

-u

Disables ANSI binds.

-v

Displays the precompiler version information only (without precompiling).

-w

Turns off display of warning messages.

-x

Uses external configuration files. See the CS_EXTERNAL_CONFIG property described in the *Open Client Client-Library/C Reference Manual*, and the INITIALIZE_APPLICATION statement described in the *Open Client Embedded SQL/C Programmers Guide*.

-y
: Supports S_TEXT and CS_IMAGE datatypes so they can be used as input host variables. At runtime, data is directly included into the character string sent to the server. Only static SQL statements are supported; use of text and image as input parameters to dynamic SQL is not supported. This substitution of arguments into command strings is performed only if the -y command line option is used.

filename[.ext]
: Specifies the input file name of the ESQL/C source program. The file name format and length can be anything that complies with the applicable rules.

Examples

Example 1 Runs the precompiler (ANSI-compliant):

```
cpre program.pc
```

Example 2 Runs the precompiler with generated stored procedures and FIPS flagging (ANSI-compliant):

```
 cpre -G -f program1.pc
```

Example 3 Runs the precompiler for input file with cursors open across transactions (not ANSI-compliant):

```
cpre -a program1.pc
```

Example 4 Displays the precompiler version information only:

```
cpre -v
```

Example 5 Runs the precompiler with the highest level of SQL checking:

```
cpre -K SEMANTIC -Uuser_id -Ppassword -Sserver_name -Dpubs2 example1.pc
```

Usage

- The cpre command defaults are set up for ANSI standard behavior.
- The -a, -c, -f, -m, -r, and -V options affect only the connect statement. If your source file does not contain a connect statement, or if you use -e or -x , these options have no effect.
- Options work with or without a space before the argument.
  For example, either of these formats works:

   -T dbg
   or
  -T dbg

- The precompiler can handle multiple input files. However, you cannot use the option -O *target_file_name*, but must accept the default target file names (see “Target file” above). If you use -G[isql_file_name], you cannot specify an argument; the default isql file names are *first_input_file.sql*, *second_input_file.sql*, and so on. If you use option -L[listing_file_name], you cannot specify an argument; the default listing file names are *first_input_file.lis, second_input_file.lis*, and so on.
- By default, cpre generate calls to ct_options that enable ANSI-style binding of indicator variables (CS_ANSI_BINDS). If indicator variables for nullable host variables (*columns*) are not available, Client-Library generates a fatal runtime error and aborts the application in use. You can avoid these issues by using -u with cpre. You can also disable ANSI binds by setting CS_ANSI_BINDS to cs_false in the *ocs.cfg* file.

Developing an application

This section lists the steps most commonly used in developing an Embedded SQL application. You may need to adapt this process to meet your own requirements. Perform these steps at the DOS command prompt.

1. Run the precompiler with options -c, -Ddatabase_name, -Ppassword, -Sserver_name, -K[SYNTAX| SEMANTIC], and -Uuser_id for syntax checking and debugging. Do not use -G[isql_file_name]. Compile and link the program to make sure the syntax is correct.
2. Make all necessary corrections. Run the precompiler with options -Ddatabase_name, -G[isql_file_name], and -Ttag_id to generate stored procedures with tag IDs for a test program. Compile and link the test program. Load the stored procedures using:

```
isql -Ppassword -Sserver_name -Uuser_id -Gisql_file_name
```

3. Run tests on your program.
4. Run the precompiler with options -Ddatabase_name and -G[isql_file_name] (but without option -T) on the corrected version of the program. Compile and link the program. Load the stored procedures using:

```
isql -Ppassword -Sserver_name -Uuser_id -Gisql_file_name
```

The final distribution program is ready to run.

How precompilers determine the names of their servers

You can connect with an Adaptive Server at precompile time, which allows you to do additional syntax checking at that time. The precompiler determines the name of its server in one of three ways:

- Using the -S option on the cpre command line

- Setting the DSQUERY variable
- Using the default value, “SYBASE”

The -S option overrides the value set by DSQUERY.

To specify a server on the precompile command line, use:

```
cpre -Usa -P -Sserver_name
```

As an alternative, you can omit the server name from the connection call or statement, and *server_name* will take its value from the runtime value of the DSQUERY environment variable. If the application user has not set DSQUERY, the runtime value for the server name defaults to “SYBASE.” See the *Open Client and Open Server Configuration Guide for UNIX* for more information about DSQUERY.

cpre defaults

Table A-6 lists the options and defaults for the cpre and cobpre utilities:

Table A-6: cpre and cobpre defaults

Option	**Default if option not used**
-C *compiler*	The mf_byte compiler for COBOL. ANSI-C for C.
-D *database_name*	The default database on Adaptive Server.
-F *fips_level*	(No FIPS flags available.)
-G [*isql_file_name*]	No stored procedures are generated.
-I *include_path_name*	Default directory is the */include* directory of the Sybase release directory.
-J *charset_locale_name*	[platform-specific]
-K [syntax | semantic | none]	If neither syntax nor semantic is selected, the default setting is “None.”
-L [*listing_file_name*]	No listing file is generated.
-N *interface_file_name*	The *interfaces* file in the Sybase release directory.
-O *target_file_name*	The default target file name is the input file name with the extension *.cbl* or *.c* appended (or replacing any input file name extension).
-P *password*	You are not prompted for a password unless you use -U*user_id*.
-S server_name	The default Adaptive Server name is taken from the DSQUERY environment variable.
-T *tag_id*	No tag IDs are added to the stored procedure names generated with -G.
-U *user_id*	None.
-V *version_number*	CS_VERSION_125 for version 12.5.x CS_VERSION_150 for version 15.0 CS_VERSION_155 for version 15.5
-Z *language_locale_name*	[platform/environment specific]

cobpre

Description

cobpre precompiles a COBOL source program to produce target, listing, and isql files. This utility is in the *$SYBASE/$SYBASE_OCS/bin*.

Note The precompilers for 64-bit applications are cobpre64 and cobpre_r64. The cobpre64 precompiler is the non-reentrant precompiler, and the cobpre_r64 is the reentrant version

Syntax

```
cobpre
   [-Ccompiler]
   [-Ddatabase_name]
   [-Ffips_level]
   [-G[isql_file_name]]
   [-Iinclude_path_name]
   [-Jcharset_locale_name]
   [-Ksyntax_level]
   [-L[listing_file_name]]
   [-M]
   [-Ninterface_file_name]
   [-Otarget_file_name]
   [-Ppassword]
   [-Sserver_name]
   [-Ttag_id]
   [-Uuser_id]
   [-Vversion_number]
   [-Zlanguage_locale_name]
   [@ options_file]
   [-a] [-b] [-c] [-d] [-e] [-f] [-l] [-m] [-r] [-s] [-u] [-v] [-w] [-x] [-y]
   filename[.ext]
```

Note You can use either a slash (/) or a dash (-) to specify options; for example, cobpre -l and cobpre /l are equivalent.

Parameters

-C *compiler*
Specifies the target host language compiler values, such as:

- "mf_byte" – Micro Focus COBOL with byte-aligned data (-C NOIBMCOMP).
- "mf_word" – Micro Focus COBOL with word-aligned data (-C IBMCOMP).

-D *database_name*
Specifies the name of the database to parse against. Use this option to check SQL semantics at precompile time. If you also specify -G, is specified, a use *database* command is added to the beginning of the *filename.sql* file. If you do not use this option, the precompiler uses the default database on the Adaptive Server.

-F *fips_level*
Checks for the specified conformance level. Currently, the precompiler can check for SQL89 or SQL92E.

-G *isql_file_name*

Generates stored procedures for appropriate SQL statements and saves them to a file for input to the database through isql. isql_file_name is optional. If you have multiple input files, you may use -G, but you cannot specify an argument.

If you have multiple input files or do not specify the argument, the default target file names are the input file names with the extension *.isql* either appended or replacing any existing input file name extension.

Also, see option -T tag_id to specify tag IDs for stored procedures.

If you do not use the -G option, no stored procedures are generated.

-I *include_path_name*

Specifies a directory complete with the path name, where Embedded SQL searches for *include* files. You can specify this option any number of times. Embedded SQL searches the directories in commandline order. If you do not use this option, the default is the */include* directory of the Sybase release directory and the current working directory.

-J *charset_locale_name*

Specifies the character set of the source file that is being precompiled. The option’s value must be a locale name that corresponds to an entry in the locales file. If you do not specify -J, the precompiler interprets the source file as being in the precompiler’s default character set.

To determine which character set to use as the default, the precompiler looks for a locale name. CS-Library searches for the information in the this order:

1. LC_ALL
2. LANG

   If LC_ALL is defined, CS-Library uses its value as the locale name. If LC_ALL is not defined but LANG is defined, CS-Library uses its value as the locale name. If none of these locale values are defined, CS-Library uses a locale name of “default.”

The precompiler looks up the locale name in the *locales.dat* file and uses the character set associated with the locale name as the default character set.

-K *syntax_level*
Specifies the level of syntax checking to perform:

- none (default)
- syntax
- semantic

If you use either syntax or semantic, you must also specify the -U, -P, -S, and -D options that Embedded SQL can connect to your Adaptive Server.

If you do not use this option, the precompiler does not connect to a server or perform SQL syntax checking of the input file beyond what is required to generate the target file.

-L *listing_file_name*
Generates one or more listing files. *listing_file_name*, which is an optional argument, is a version of the input file that numbers each line, and includes any applicable error messages. If you have multiple input files, you may use -L, but you cannot specify an argument.

If you have multiple input files or do not specify the argument, the default listing file names are the input file names with the extension *.lis* either appended, or replacing any existing input file name extension.

If you do not use this option, no listing file is generated.

-M
Turns on security feature. Sets B1 secure labels.

-N *interface_file_name*
Specifies the configuration file name, *interfaces*, to the precompiler.

-O *target_file_name*
Specifies the target or output file name. If you have multiple input files, you cannot use this option; default target file names are assigned—the input file name with the extension *.cbl* either appended, or replacing any existing input file name extension.

-P *password* (used with option -Uuser_id)
Specifies an Adaptive Server password for SQL syntax checking at precompile time. Using -P without an argument, or with the keyword NULL as an argument, specifies a null ("") password. If you use option -Uuser_id without using -P, the precompiler prompts you to enter a password. Must be used with the -G flag.

-S *server_name*

Specifies the name of the Adaptive Server for SQL syntax checking at precompile time. If you do not use this option, the default Adaptive Server name is taken from the DSQUERY environment variable. If DSQUERY is not set, SYBASE is used as the name of the server.

-T *tag_id* (used with option -G)

Specifies a tag ID (up to 3 characters) to append to the end of the generated stored procedure group name.

For example, if you enter -Tdbg as part of your command, generated stored procedures are assigned the name of the input file with the tag ID *dbg* appended: *program_dbg;1*, *program_dbg;2,* and so on.

Programmers can use tag IDs to test changes to an existing application without destroying the existing generated stored procedures, which may be in use.

If you do not use this option, no tag ID is added to the stored procedure name.

-U *user_id*

Specifies the Adaptive Server user ID. This option allows you to check SQL syntax at precompile time. It causes the precompiler to pass SQL statements to the server for parsing only. If the server detects syntax errors, the errors are reported and no code is generated. If you are not using -P*password,* this option prompts you to enter a password.

Also see -K, -P, -S, and -D options.

-V *version_number*

Specifies the Client-Library version number. This must match one of the values from cobpub.cbl. If you do not use this option, the default is the most recent version of Client-Library available with the precompiler (CS_VERSION_155 for Open Client and Open Server version 15.5).

-Z *language_locale_name*
Specifies the language and character set that the precompiler uses for messages. If you do not specify -Z, the precompiler uses its default language and character set for messages.

To determine which language and character set to use as its default for messages, the precompiler performs the following, in order:

1. Looks for a locale name. CS-Library searches for the information first in LC_ALL, then in LANG. If neither of these locale values are defined, CS-Library uses a locale name of “default.”
2. Looks up the locale name in the *locales.dat* file to determine which language and character set are associated with it.
3. Loads localized messages and character set information appropriate to the language and character set determined in step 2.

@options_file
Can be used to specify a file containing any of the above command-line arguments. The precompiler reads the arguments contained in this file in addition to any arguments already specified. If the file specified with *@options_file* contains names of the files to precompile, place the argument at the end of the command line.

-a
Allows cursors to remain open across transactions. If you do not use this option, cursors behave as though set close on endtran on were in effect. This behavior is ANSI-compatible. See the *Adaptive Server Enterprise Reference Manual* for information about cursors and transactions.

-b
Disables rebinding of host variable addresses typically used in fetch statements. If you do not use this option, a rebind occurs on every fetch statement unless you specify otherwise in your Embedded SQL/C program.

The -b option differs in the 11.1 and 10.x versions of the Embedded SQL precompilers:

- For the 11.1 and later versions of cobpre, the norebind attribute applies to all fetch statements of a cursor for which the declaration was precompiled with the -b option.
- For the 10.0 and earlier versions of cobpre, the norebind attribute applied to all fetch statements in each Embedded SQL source file precompiled with -b, regardless of where the cursors were declared.

-c

Turns on the debugging feature of Client-Library by generating calls to ct_debug.

This option is useful during application development but should be turned off for final application delivery. The application must be linked and run with the libraries and DLLs located in the *%SYBASE%\%SYBASE_OCS\devlib* directory of the Sybase release directory.

-d

Turns off delimited identifiers (identifiers delimited by double quotes) and allows quoted strings in SQL statements to be treated as character literals.

-e

When processing an exec sql connect statement, directs Client-Library to use the external configuration file to configure the connection. Also see the -x option and CS_CONFIG_BY_SERVERNAME property in the *Open Client Client-Library/C Reference Manual*.

Without this option, the precompiler generates Client-Library function calls to configure the connection. See the *Open Client Client-Library/C Reference Manual* for information about the external configuration file

-f

Turns on the FIPS flagger for ANSI FIPS compliance checking.

-l

Turns off generation of #line directives.

-m

Runs the application in Sybase auto-commit mode, which means that transactions are not chained. Explicit begin and end transactions are required, or every statement is immediately committed. If you do not specify this option, the application runs in ANSI-style chained transaction mode.

-r

Disables repeatable reads. If you do not use this option, a set transaction isolation level 3 statement, which executes during connect statements, is generated. The default isolation level is 1.

-s

Includes static function declarations.

-u

Disables ANSI binds.

-v
: Displays the precompiler version information only (without precompiling).

-w
: Turns off display of warning messages.

-x
: Uses external configuration files. See the CS_EXTERNAL_CONFIG property described in the *Open Client Client-Library/C Reference Manual*, and the INITIALIZE_APPLICATION statement described in the *Open Client Embedded SQL/C Programmers Guide*.

-y
: Supports S_TEXT and CS_IMAGE datatypes so they can be used as input host variables. At runtime, data is directly included into the character string sent to the server. Only static SQL statements are supported; use of text and image as input parameters to dynamic SQL is not supported. This substitution of arguments into command strings is performed only if the -y commandline option is used.

filename[.ext]
: Specifies the input file name of the ESQL/C source program. The file name format and length can be anything that complies with the applicable rules.

Examples

Example 1 Runs the precompiler (ANSI-compliant):

```
cobpre program.pco
```

Example 2 Runs the precompiler with generated stored procedures and FIPS flagging (ANSI-compliant):

```
cobpre -G -f program1.pco
```

Example 3 Runs the precompiler for input file with cursors open across transactions (not ANSI-compliant):

```
cobpre -a program1.pco
```

Example 4 Displays the precompiler version information only:

```
cobpre -v
```

Example 5 Runs the precompiler with the highest level of SQL checking:

```
cobpre -K SEMANTIC -Uuser_id -Ppassword -Sserver_name \
      -Dpubs2 example1.pco
```

Usage

- The cobpre| command defaults are set up for ANSI standard behavior.

- Target file:
  The default target file name is the input file name with the extension *.cbl* (for Micro Focus COBOL) appended (or replacing any input file name extension). If you have only one input file, you may use option -O target_file_name to specify a target file name. If you have multiple input files, the default target files will be named *first_input_file.cbl*, *second_input_file.cbl*, etc.
- Options work with or without a space before the argument.
  For example, either of these formats works:

   -T dbg
   or
  -T dbg

- The precompiler can handle multiple input files. However, you cannot use the option -O *target_file_name*, but must accept the default target file names (see "Target file" above). If you use -G[isql_file_name], you cannot specify an argument; the default isql file names are *first_input_file.sql*, *second_input_file.sql*, and so on. If you use -L[listing_file_name], you cannot specify an argument; the default listing file names are *first_input_file.lis*, *second_input_file.lis*, and so on.
- By default, cobpre generate calls to ct_options that enable ANSI-style binding of indicator variables (CS_ANSI_BINDS). If indicator variables for nullable host variables (*columns*) are not available, Client-Library generates a fatal runtime error and aborts the application in use. You can avoid these issues by using -u with cobpre. You can also disable ANSI binds by setting CS_ANSI_BINDS to cs_false in the *ocs.cfg* file.

Developing an application

This section lists the steps most commonly used in developing an Embedded SQL application. You may need to adapt this process to meet your own requirements. Perform these steps at the DOS command prompt.

1. Run the precompiler with options -Ddatabase_name, -Ppassword, -Sserver_name, -K[SYNTAX| SEMANTIC], and -Uuser_id for syntax checking and debugging. Do not use -G[isql_file_name]. Compile and link the program to make sure the syntax is correct.
2. Make all necessary corrections. Run the precompiler with options -Ddatabase_name, -G[isql_file_name], and -Ttag_id to generate stored procedures with tag IDs for a test program. Compile and link the test program. Load the stored procedures using:

```
isql -Ppassword -Sserver_name -Uuser_id -Gisql_file_name
```

3. Run tests on your program.
4. Run the precompiler with options -Ddatabase_name and -G[isql_file_name] (but without option -T) on the corrected version of the program. Compile and link the program. Load the stored procedures using:

```
isql -Ppassword -Sserver_name -Uuser_id -Gisql_file_name
```

The final distribution program is ready to run.

How precompilers determine the names of their servers

You can connect with an Adaptive Server at precompile time, which allows you to do additional syntax checking at that time. The precompiler determines the name of its server in one of three ways:

- Using the -S option on the cpre or cobpre command line
- Setting the DSQUERY variable
- Using the default value, "SYBASE"

The -S option overrides the value set by DSQUERY.

To specify a server on the precompile command line, use:

```
cobpre -Usa -P -Sserver_name
```

As an alternative, you can omit the server name from the connection call or statement, and *server_name* then takes its value from the runtime value of the DSQUERY environment variable. If the application user has not set DSQUERY, the runtime value for the server name defaults to "SYBASE." See the *Open Client and Open Server Configuration Guide for UNIX* for more information about DSQUERY.

cobpre | cpre defaults

See Table A-6 on page 121 for a list of options and defaults for the cpre and cobpre utilities.

defncopy

Description

Copies definitions for specified views, rules, defaults, triggers, or procedures from a database to an operating system file or from an operating system file to a database. This utility is in the *$SYBASE/$SYBASE_OCS/bin*.

Note defncopy cannot copy table definitions or reports created with Report Workbench™.

Syntax

```
defncopy
   [-a display_charset]
   [-I interfaces_file]
   [-J [client_charset]]
   [-P password]
   [-R remote_server_principal]
   [-S [server_name]]
   [-U user_name]
   [-v]
   [-V [security_options]]
   [-X]
   [-z language]
   [-Z security_mechanism]
   {in file_name database_name |  out file_name database_name
   [owner.]object_name [[owner.]object_name...] }
```

Parameters

-a *display_charset*

Runs defncopy from a terminal where the character set differs from that of the machine on which defncopy is running. Use -a with -J to specify the character set translation file (*.xlt* file) required for the conversion. Use -a without -J only if the client character set is the same as the default character set.

Note The ascii_7 character set is compatible with all character sets. If either the Adaptive Servers or the client's character set is set to ascii_7, any 7-bit ASCII character is allowed to pass unaltered between client and server. Other characters produce conversion errors. Character set conversion issues are discussed thoroughly in the *Adaptive Server Enterprise System Administration Guide*.

-I *interfaces_file*

Specifies the name and location of the interfaces file to search when connecting to Adaptive Server. If you do not specify -I, defncopy looks for the interfaces file, *interfaces*, located in the Sybase release directory.

-J *client_charset*
Specifies the character set to use on the client. A filter converts input between *client_charset* and the Adaptive Server character set.

-J *client_charset* requests that Adaptive Server convert to and from *client_charset*, the client's character set.

-J with no argument sets character set conversion to NULL. No conversion takes place. Use this if the client and server are using the same character set.

Omitting -J sets the character set to a default for the platform.The default may not necessarily be the character set that the client is using.

-P *password*
Allows you to specify your password. If you do not specify -P, defncopy prompts for your password. This option is ignored if -V is used.

-R *remote_server_principal*
Specifies the principal name for the server. By default, a server's principal name matches the server's network name (which is specified with the -S option or the DSQUERY environment variable). The -R parameter must be used when the server's principal name and network name are not the same.

-S *server_name*
Specifies the name of the Adaptive Server to connect to. If you specify -S with no argument, defncopy looks for a server named SYBASE. If you do not specify -S, defncopy uses the server specified by your DSQUERY environment variable.

-U *user_name*
Allows you to specify a login name. Login names are case-sensitive. If you do not specify *username*, defncopy uses the current user's operating system login name.

-v

Displays the version number and copyright message of defncopy, then returns to the operating system.

SDK binaries like defncopy use the same names in both 32-bit and 64-bit products. Installing Adaptive Server, the SDK, or Open Server 64-bit products with other Sybase 32-bit products overwrites the 32-bit binaries. Starting with Adaptive Server 15.0.2 and SDK/Open Server 15.0 ESD #9, the 64-bit binaries are replaced with 32-bit binaries on all 64-bit UNIX platforms. Since 32-bit binaries are included in the 64-bit EBF, the -v option of defncopy is no longer a valid way to check the EBF number for 64-bit products. Instead, use the UNIX strings and grep commands to confirm the EBF numbers for both Open Client and Open Server.

For example, to find the string containing the EBF number in the *libsybct64.a* library, enter:

```
strings -a libsybct64.a | grep Sybase
```

This returns a string similar to:

```
Sybase Client-Library/15.5/P/DRV.15.5.0/SPARC/Solaris
8/BUILD1550-001/64bit/OPT/Mon Aug 10 23:04:17 2009
```

To find the string containing the EBF number in the *libsybsrv64.a* library, enter:

```
strings -a libsybsrv64.a | grep Sybase
```

This returns a string similar to:

```
Sybase Server-Library/15.5/P/DRV.15.5.0/SPARC/Solaris
8/BUILD1550-001/64bit/OPT/Mon Aug 10 23:06:27 2009
```

-V *security_options*

Specifies network-based user authentication. With this option, the user must log in to the network's security system before running defncopy. In this case, users must supply their network user name with the -U parameter; any password supplied with the -P parameter is ignored.

-V can be followed by a *security_options* string that enables additional security services:

- c – enable data confidentiality service.
- i – enable data integrity service.
- m – enable mutual authentication for connection establishment.
- o – enable data origin stamping service.
- q – enable out-of-sequence detection.
- r – enable data replay detection.

-X

Specifies that in this connection to the server, the application initiate the login with client-side password encryption. defncopy (the client) specifies to the server that password encryption is desired. The server sends back an encryption key, which defncopy uses to encrypt your password, and the server uses the key to authenticate your password when it arrives.

If defncopy fails, the system creates a core file which contains your password. If you did not use the encryption option, the password appears in plain text in the file. If you used the encryption option, your password is not readable.

-z *language*

The official name of an alternate language that the server uses to display defncopy prompts and messages. Without the -z flag, defncopy uses the server's default language.

Add languages to an Adaptive Server during installation, or afterwards using the utility langinst or the stored procedure sp_addlanguage.

-Z *security_mechanism*

Specifies the name of a security mechanism to use on the connection.

Security mechanism names are defined in the *libtcl.cfg* configuration file located in the *$SYBASE/ini* directory. If no *security_mechanism* name is supplied, the default mechanism is used. See the description of the *libtcl.cfg* file in the *Open Client and Open Server Configuration Guide for UNIX*.

in | out
: Specifies the direction of definition copy.

file_name
: Specifies the name of the operating system file destination or source for the definition copy. The copy out overwrites any existing file.

database_name
: Specifies the name of the database to copy the definitions to or from.

object_name
: Specifies names of database objects for defncopy to copy out. Do not use *object_name* when copying definitions in.

owner
: Specifying *owner* is optional if you or the database owner own the table being copied. If you do not specify an owner, defncopy first looks for a table of that name that you own, and then looks for one owned by the database owner. If another user owns the table, you must specify the owner name or the command fails.

Examples

Example 1 Copies definitions from the file *new_proc* into the database stagedb on server MERCURY. The connection with MERCURY is established with a user of name "sa" and a NULL password.

```
defncopy -Usa -P -SMERCURY in new_proc stagedb
```

Example 2 Copies definitions for objects sp_calccomp and sp_vacation from the employees database on the Sybase server to the file *dc.out*. Messages and prompts appear in French. The user is prompted for a password.

```
defncopy -S -z french out dc.out employees sp_calccomp sp_vacation
```

Usage

- Invoke the defncopy program directly from the operating system. defncopy provides a noninteractive to copy out definitions (create statements) for views, rules, defaults, triggers, or procedures from a database to an operating system file. Alternatively, it copies in all the definitions from a specified file.

  You must have select permission on the sysobjects and syscomments tables to copy out definitions; you do not need permission on the object itself.

- You must have the appropriate create permission for the type of object you are copying in. Objects copied in belong to the copier. A system administrator copying in definitions on behalf of a user must log in as that user to give the user proper access to the reconstructed database objects.

- The in *filename* or out *filename* and the database name are required and must be unambiguously stated. For copying out, use file names that reflect both the object's name and its owner.
- defncopy ends each definition that it copies out with:

  ```
  /* ### DEFNCOPY: END OF DEFINITION  */
  ```

  When assembling definitions in an operating system file to be copied into a database using defncopy, each definition must be terminated using the "END OF DEFINITION" string.
- Enclose values specified to defncopy in quotation marks if they contain characters that could be significant to the shell.

Warning! Long comments (more than 100 characters) placed before a create statement may cause defncopy to fail.

isql

Description

Interactive SQL parser to Adaptive Server. This utility is in *$SYBASE/$SYBASE_OCS/bin*.

Syntax

isql [-b] [-e] [-F] [-n] [-p] [-v] [-W] [-X] [-Y] [-Q]
[-a *display_charset*]
[-A *packet_size*]
[-c *cmdend*]
[-D *database*]
[-E *editor*]
[-h *header*]
[-H *hostname*]
[-i *inputfile*]
[-I *interfaces_file*]
[-J *client_charset*]
[-K *keytab_file*]
[-l *login_timeout*]
[-m *errorlevel*]
[-M*LabelName LabelValue*]
[-o *outputfile*]
[-P *password*]
[-R *remote_server_principal*]
[-s *col_separator*]
[-S *server_name*]
[-t *timeout*]
[-U *username*]

[--URP *remotepassword*
[-V [*security_options*]]
[-w *column_width*]
[-x *trusted.txt_file*]
[-y *sybase_directory*]
[-z *localename*]
[-Z *security_mechanism*]
[--appname "*application_name*"]
[--conceal [':?' | '*wildcard*']]
[--help]
[--history [p]*history_length* [--history_file *history_filename*]]
[--retserverror]

Parameters

-a *display_charset*

Allows you to run isql from a terminal where the character set differs from that of the machine on which isql is running. Use -a with -J to specify the character set translation file (*.xlt* file) required for the conversion. Use -a without -J only if the client character set is the same as the default character set.

Note The ascii_7 character set is compatible with all character sets. If either the Adaptive Servers or the client's character set is set to ascii_7, any 7-bit ASCII character is allowed to pass unaltered between client and server. Other characters produce conversion errors. Character set conversion issues are discussed thoroughly in the *Adaptive Server Enterprise System Administration Guide*.

-A *packet_size*

Specifies the network packet size to use for this isql session. For example, the following sets the packet size to 4096 bytes for the isql session:

```
isql -A 4096
```

To check your network packet size, enter:

```
select * from sysprocesses
```

The value appears under the *network_pktsz* heading.

packet_size must be between the values of the default network packet size and maximum network packet size configuration variables, and must be a multiple of 512.

Use larger-than-default packet sizes to perform I/O-intensive operations, such as readtext or writetext operations.

Setting or changing the Adaptive Server packet size does not affect remote procedure calls' packet size.

-b
Disables the display of the table headers output.

-c *cmdend*
Resets the command terminator. By default, you can terminate commands and send them to Adaptive Server by typing "go" on a line by itself. When you reset the command terminator, do not use SQL reserved words or control characters. Escape shell metacharacters such as , ? ( ) [ ] $ and so on.

-D *database*
Selects a database in which the isql session begins.

-e
Echoes input.

-E *editor*
Specifies an editor other than your default editor (such as vi). To invoke it, enter its name as the first word of a line in isql.

-F
Enables the FIPS flagger. When you specify the -F parameter, the server returns a message when it encounters a nonstandard SQL command. This option does not disable SQL extensions. Processing completes when you issue the non-ANSI SQL command.

-h *header*
Specifies the number of rows to print between column headings. The default prints headings only once for each set of query results.

-H *hostname*
Sets the client host name.

-i *inputfilename*
Specifies the name of an operating system file to use for input to isql. The file must contain command terminators ("go" by default).

- Specifying the parameter as follows is equivalent to < *inputfile*:

  ```
  -i inputfile
  ```

- If you use -i and do not specify your password on the command line, isql prompts you for it.
- If you use < *inputfile* and do not specify your password on the command line, you must specify your password as the first line of the input file.

-I *interfaces_file*
: Specifies the name and location of the interfaces file to search when connecting to Adaptive Server. If you do not specify -I, isql looks for an interfaces file, *interfaces* located in the Sybase release directory.

-J *client_charset*
: Specifies the character set to use on the client. -J*client_charset* requests that Adaptive Server convert to and from *client_charset*, the character set used on the client. A filter converts input between *client_charset* and the Adaptive Server character set.

  -J with no argument sets character set conversion to NULL. No conversion takes place. Use this if the client and server use the same character set.

  Omitting -J sets the character set to a default for the platform. The default may not necessarily be the character set that the client is using. For more information about character sets and the associated flags, see the *Adaptive Server Enterprise System Administration Guide*.

-K *keytab_file*
: Used only with DCE security. *keytab_file* specifies a DCE keytab file that contains the security key for the user name specified with -U option. Create keytab files using the DCE dcecp utility. See your DCE documentation.

  If -K is not supplied, the bcp user must be logged in to DCE with the same user name as specified with the -U option.

-l *login_timeout*
: Specifies the maximum timeout value allowed when connecting to Adaptive Server. The default is 60 seconds. This value affects only the time that isql waits for the server to respond to a login attempt. To specify a timeout period for command processing, use the -t *timeout* parameter.

-m *errorlevel*
: Customizes the error message display. For errors of the severity level specified or higher, only the message number, state, and error level appear; no error text appears. For error levels lower than the specified level, nothing appears.

-M *LabelName LabelValue*
(Secure SQL Server only) enables multilevel users to set the session labels for the bulk-copy. Valid values for *LabelName* are:

- curread (current read level) is the initial level of data that you can read during this session. curread must dominate curwrite.
- curwrite (current write level) is the initial sensitivity level that is applied to any data that you write during this session.
- maxread (maximum read level) is the maximum level at which you can read data. This is the upper bound to which you as a multilevel user can set curread during the session. maxread must dominate maxwrite.
- maxwrite (maximum write level) is the maximum level at which you can write data. This is the upper bound to which you as a multilevel user can set curwrite during a session. maxwrite must dominate minwrite and curwrite.
- minwrite (minimum write level) is the minimum level at which you can write data. This is the lower bound to which you as a multilevel user can set curwrite during a session. minwrite must be dominated by maxwrite and curwrite.

LabelValue is the actual value of the label, expressed in the human-readable format used on your system (for example, "Company Confidential Personnel").

-n
Removes numbering and the prompt symbol (>) from echoed input lines in the output file when used with -e.

-o *output_filename*
Specifies the name of an operating system file to store the output from isql. Specifying the parameter as -o *outputfile* is similar to > *outputfile*.

-p
Prints performance statistics.

-P *password*
specifies your current Adaptive Server password. This option is ignored if -V is used. Passwords are case-sensitive and can be 6 – 30 characters in length. If your password is NULL, use -P without any password.

-Q
Provides clients with failover capability. See the *Adaptive Server Enterprise Using Sybase Failover in a High Availability System.*

-R *remote_server_principal*
: Specifies the principal name for the server. By default, a server’s principal name matches the server’s network name (which is specified with the -S option or the DSQUERY environment variable). Use -R when the server’s principal name and network name are not the same.

-s *colseparator*
: Resets the column separator character, which, by default, is blank. To use characters that have special meaning to the operating system (for example, “|”, “;”, “&”, “<“, “>”), enclose them in quotes or precede them with a backslash.

The column separator appears at the beginning and the end of each column of each row.

-S *server*
: Specifies the name of the Adaptive Server to connect to. isql looks this name up in the interfaces file. If you specify -S with no argument, isql looks for a server named SYBASE. If you do not specify -S, isql looks for the server specified by your DSQUERY environment variable.

-t *timeout*
: Specifies the number of seconds before a SQL command times out. If you do not specify a timeout, a command runs indefinitely. This affects commands issued from within isql, not the connection time. The default timeout for logging in to isql is 60 seconds.

-U *username*
: Specifies a case-sensitive login name.

--URP *remotepassword*
: Enables setting the universal remote password *remotepassword* for clients accessing Adaptive Server.

-V *security_options*

Specifies network-based user authentication. With this option, the user must log in to the network's security system before running isql. In this case, users must supply their network user name with the -U option; any password supplied with the -P option is ignored.

-V can be followed by a *security_options* string that enables additional security services:

- c – enable data confidentiality service.
- d – enable credential delegation and forward the client credentials to the gateway application.
- i – enable data integrity service.
- m – enable mutual authentication for connection establishment.
- o – enable data origin stamping service.
- q – enable out-of-sequence detection.
- r – enable data replay detection.

-v

Prints the version and copyright message of the isql and then exits.

SDK binaries like isql have the same names in both the 32-bit and 64-bit products. Installing Adaptive Server, the SDK, or Open Server 64-bit products with other Sybase 32-bit products overwrites the 32-bit binaries. Starting with Adaptive Server 15.0.2 and SDK/Open Server 15.0 ESD #9, the 64-bit binaries are replaced with 32-bit binaries on all 64-bit UNIX platforms. Since 32-bit binaries are included in the 64-bit EBF, the -v option of isql is no longer a valid way to check the EBF number for 64-bit products. Instead, use the UNIX strings and grep commands to confirm the EBF numbers for both Open Client and Open Server.

For example, to find the string containing the EBF number in the *libsybct64.a* library, enter:

```
strings -a libsybct64.a | grep Sybase
```

This returns a string similar to:

```
Sybase Client-Library/15.5/P/DRV.15.5.0/SPARC/Solaris
8/BUILD1550-001/64bit/OPT/Mon Aug 10 23:04:17 2009
```

To find the string containing the EBF number in the *libsybsrv64.a* library, enter:

```
strings -a libsybsrv64.a | grep Sybase
```

This returns a string similar to:

```
Sybase Server-Library/15.5/P/DRV.15.5.0/SPARC/Solaris
8/BUILD1550-001/64bit/OPT/Mon Aug 10 23:06:27 2009
```

-w *columnwidth*
sets the screen width for output. The default is 80 characters. When an output line reaches its maximum screen width, it breaks into multiple lines.

-W
Specifies that if the server to which isql is attempting to connect supports neither normal password encryption nor extended password encryption, plain text password retries are disabled. If this option is used, the CS_SEC_NON_ENCRYPTION_RETRY connection property is set to CS_FALSE, and plain text (unencrypted) passwords will not be used in retrying the connection.

-x *trusted.txt_file*
Specifies an alternate *trusted.txt* file.

-X
Initiates the login connection to the server with client-side password encryption. isql (the client) specifies to the server that password encryption is desired. The server sends back an encryption key, which isql uses to encrypt your password, and the server uses the key to authenticate your password when it arrives.

This option can result in normal or extended password encryption, depending on connection property settings at the server. If CS_SEC_ENCRYPTION is set to CS_TRUE, normal password encryption is used. If CS_SEC_EXTENDED_ENCRYPTION is set to CS_TRUE, extended password encryption is used. If both CS_SEC_ENCRYPTION and CS_SEC_EXTENDED_ENCRYPTION are set to CS_TRUE, extended password encryption takes precedence.

If isql fails, the system creates a core file that contains your password. If you did not use the encryption option, the password appears in plain text in the file. If you used the encryption option, your password is not readable.

-y *sybase_directory*
Sets an alternate Sybase home directory.

-Y
Tells the Adaptive Server to use chain transactions.

-z *localename*

The official name of an alternate language to display isql prompts and messages. Without -z, isql uses the server's default language. Add languages to an Adaptive Server during installation, or afterward using the utility langinst or the sp_addlanguage stored procedure.

-Z *security_mechanism*

Specifies the name of a security mechanism to use on the connection.

Security mechanism names are defined in the *libtcl.cfg* configuration file located in the *$SYBASE/ini* directory. If no *security_mechanism* name is supplied, the default mechanism is used. See the description of the *libtcl.cfg* file in the *Open Client and Open Server Configuration Guide for UNIX*.

--appname "*application_name*"

Allows you to change the default application name *isql* to the isql client application name. This simplifies:

- Testing of Adaptive Server cluster routing rules for incoming client connections based on the client application name.
- Switching between alternative settings for isql in *$SYBASE/$SYBASE_OCS/config/ocs.cfg*, such as between debugging and normal sessions.
- Identification of the script that started a particular isql session from within Adaptive Server.

application_name is the client application name. You can retrieve the client application name from sysprocesses.program_name after connecting to your host server.

application_name has a maximum length of 30 characters. You must enclose the entire application name in single quote or double quote characters if it contains any white spaces that do not use the backslash escape character. You can set the *application_name* to an empty string.

Note You can also set the client application name in *ocs.cfg* using the CS_APPNAME property.

--conceal [':?' | '*wildcard*']

Hides your input during an isql session. The --conceal option is useful when entering sensitive information, such as passwords.

wildcard, a 32-byte variable, specifies the character string that triggers isql to prompt you for input during an isql session. For every wildcard that isql reads, isql displays a prompt that accepts your input but does not echo the input to the screen. The default wildcard is :?.

Note --conceal is silently ignored in batch mode.

See "Using prompt labels and double wildcards in an isql session" on page 155.

--help

Displays a brief description of syntax and usage for the isql utility consisting of a list of available arguments.

--history [p]*history_length* [--history_file *history_filename*]

Loads the contents of the command history log file, if it exists, when isql starts. By default, the command history feature is off. Use --history command line option to activate it.

- p – indicates command history persistence; in-memory command history is saved to disk when isql shuts down. If you do not use the p option, the command history log is deleted after its contents are loaded into memory.
- *history_length* – this parameter, which is required if you use --history, is the number of commands that isql can store in the command history log. The maximum value of *history_length* is 1024; if a larger value is specified, isql silently truncates it to 1024.
- --history_file *history_filename* – indicates that isql must retrieve the command history log from *history_filename*. If p is specified, isql also uses *history_filename* to store the current session's command history. *history_filename* can include an absolute or a relative path to the log file. A relative path is based on the current directory. If you do not indicate a path, the history log is saved in the current directory.

  When --history_file is not specified, isql uses the default log file in *$HOME/.sybase/isql/isqlCmdHistory.log*:

  For information about listing, recalling, and reissuing past commands, see "Using command history" on page 155.

--retserverror

Forces isql to terminate and return a failure code when it encounters a server error of severity greater than 10. When isql encounters this type of abnormal termination, it writes the label “Msg” together with the actual Adaptive Server error number to stderr, and returns a value of 2 to the calling program. isql prints the full server error message to stdout.

Examples

Example 1 Opens a text editor where you can edit the query. When you write and save the file, you are returned to isql. The query appears; type “go” on a line by itself to execute it:

```
isql -Ujoe -Pabracadabra
 1>select *
 2>from authors
 3>where city = "Oakland"
 4>vi
```

Example 2 reset clears the query buffer. quit returns you to the operating system.

```
isql -U alma
Password:
1>select *
2>from authors
3>where city = "Oakland"
4>reset
5>quit
```

Example 3 Creates column separators using the “#” character in the output in the pubs2 database for store ID 7896:

```
isql -Usa -P -s#
1> use pubs2
2> go
1> select * from sales where stor_id = “7896”
#stor_id#ord_num                #date                            #
#-------#----------------------#--------------------------------#
#7896   #124152                #             Aug 14 1986 12:00AM#
#7896    #234518                 #               Feb 14 1991 12:00AM#



(2 rows affected)
```

Example 4 For MIT Kerberos, requests credential delegation and forwards the client credentials to MY_GATEWAY:

```
isql -Vd -SMY_GATEWAY
```

Example 5 In this retserverror example, isql returns 2 to the calling shell, prints "Msg 207" to stderr, and exits, when it encountered a server error of severity 16.

```
guest> isql -Uguest -Pguestpwd -SmyASE --retserverror
       2> isql.stderr
1> select no_column from sysobjects
2> go
Msg 207, Level 16, State 4:
Server 'myASE', Line 1:
Invalid column name 'no_column'.



guest> echo $?
2
guest> cat isql.stderr
Msg    207
guest>
```

Example 6 When you use the --help option, isql returns a brief description of syntax and usage for the isql utility consisting of a list of available arguments.

```
guest> isql --help

usage: isql [option1] [option2] ... where [options] are...
-b                 Disables the display of the table headers output.
-e                 Echoes input.
-F                 Enables the FIPS flagger.
-p                 Prints performance statistics.
-n                 Removes numbering and the prompt symbol when used
                   with -e.
-v                 Prints the version number and copyright message.
-W                 Turn off extended password encryption on connection
                   retries.
-X                 Initiates the login connection to the server with
                   client-side password encryption.
-Y                 Tells the Adaptive Server to use chained transactions.
-Q                 Enables the HAFAILOVER property.
-a display_charset Used in conjunction with -J to specify the character set
                   translation file (.xlt file) required for the conversion.
                   Use -a without -J only if the client character set is the
                   same as the default character set.
-A packet_size     Specifies the network packet size to use for this isql
                   session.
-c cmdend          Changes the command terminator.
-D database        Selects the database in which the isql session begins.
-E editor          Specifies an editor other than the default editor vi.
```

```
-h header          Specifies the number of rows to print between column
                   headings.
-H hostname        Sets the client host name.
-i inputfile       Specifies the name of the operating system file to use
                   for input to isql.
-I interfaces_file Specifies the name and location of the interfaces file.
-J client_charset  Specifies the character set to use on the client.
-K keytab_file     Specifies the path to the keytab file used for
                   authentication in DCE.
-l login_timeout   Specifies the number of seconds to wait for the server
                   to respond to a login attempt.
-m errorlevel      Customizes the error message display.
-M labelname labelvalue
                   Used for security labels. See CS_SEC_NEGOTIATE for more
                   details.
-o outputfile      Specifies the name of an operating system file to store
the output from isql.
-P password        Specifies your Adaptive Server password.
-R remote_server_principal
                  Specifies the principal name for the server as defined to
                  the security mechanism.
-s col_separator  Resets the column separator character, which is blank by
                  default.
-S server_name    Specifies the name of the Adaptive Server to which to
                  connect.
-t timeout        Specifies the number of seconds before a SQL command times
                  out.
-U username       Specifies a login name. Login names are case sensitive.
-V [security_options]
                  Specifies network-based user authentication. Valid
                  [security_options]:
                  c - Enable data confidentiality service.
                  i - Enable data integrity service.
                  m - Enable mutual authentication for connection
                      establishment.
                  o - Enable data origin stamping service.
                  q - Enable out-of-sequence detection.
                  r - Enable data replay detection.
                  d - Requests credential delegation and forwards client
                      credentials.
-w column_width   Sets the screen width for output.
-y sybase_directory
                  Sets an alternate location for the Sybase home directory.
-z localename     Sets the official name of an alternate language to display
                  isql prompts and messages.
-Z security_mechanism
```

```
                     Specifies the name of a security mechanism to use on the
                     connection.
-x trusted.txt_file Specifies an alternate trusted.txt file location.
--retserverror       Forces isql to terminate and return a failure code when it
                     encounters a server error of severity greater than 10.
--conceal [wildcard]
                     Obfuscates input in an ISQL session. The optional wildcard
                     will be used as a prompt.
```

Example 7 Sets an alternate Sybase home directory using the -y option:

```
isql -y/work/NewSybase -Uuser1 -Psecret -SMYSERVER
```

Example 8 In this example of --conceal, the password is modified without displaying the password entered. This example uses “old” and “new” as prompt labels:

```
$ isql -Uguest -Pguest -Smyase --conceal
1> sp_password
2> :? old
3> ,
4> :?:? new
5> go
old
new
Confirm new
Password correctly set.
(return status = 0)
```

Example 9 In this example of --conceal, password is modified without displaying the password entered. This example uses the default wildcard as the prompt label:

```
$ isql -Uguest -Pguest -Smyase --conceal
1> sp_password
2> :?
3> ,
4> :?:?
5> go
:?
:?
Confirm :?
Password correctly set.
(return status = 0)
```

Example 10 Activate a role for the current user. This example of --conceal uses a custom wildcard and the prompt labels “role” and “password”:

```
$ isql -UmyAccount --conceal '*'
Password:
1> set role
2> * role
3> with passwd
4> ** password
5> on
6> go
role
password
Confirm password
1>
```

Example 11 Sets the application name to “isql Session 01”:

```
isql -UmyAccount -SmyServer --appname "isql Session 01"
Password:
1>select program_name from sysprocesses
2>where spid=@@spid
3>go



program_name
-------------------
isql Session 01
```

Example 12 Sets the application name to the name of the script that started the isql session:

```
isql --appname $0
```

Example 13 This sample *ocs.cfg* file allows you to run isql normally or with network debug information. Because the configuration file is read and interpreted after the command line parameters are read and interpreted, setting CS_APPNAME to *isql* sets the application name back to isql:

```
;Sample ocs.cfg file
[DEFAULT]
;place holder

[isql]
;place holder

[isql_dbg_net]
CS_DEBUG = CS_DBG_NETWORK
CS_APPNAME = "isql"
```

To run isql normally:

```
isql -Uguest
```

To run isql with network debug information:

```
isql -Uguest --appname isql_dbg_net
```

Example 14 Loads and saves the command history using the default log file:

```
isql -Uguest -Ppassword -Smyase --history p1024
```

Example 15 Deletes *myaseHistory.log* after loading its contents to memory. The session's command history is not stored:

```
isql -Uguest -Ppassword -Smyase --history 1024
   --history_file myaseHistory.log
```

Example 16 Lists all the commands stored in the command history:

```
isql -Uguest -Ppassword -Smyase --history p1024
1> h
[1] select @@version
[2] select db_name()
[3] select @@servername
1>
```

Example 17 Lists the two most recent commands issued:

```
isql -Uguest -Ppassword -Smyase --history p1024
1> h -2
[2] select db_name()
[3] select @@servername
1>
```

Example 18 Recalls the command labeled 1 from the command history:

```
isql -Uguest -Ppassword -Smyase --history p1024
1> ? 1
1> select @@version
2>
```

Example 19 Recalls the latest issued command from the command history:

```
isql -Uguest -Ppassword -Smyase --history p1024
1> ? -1
1> select @@servername
2>
```

Usage

- Following are the commands you can use at isql prompt:

    - To terminate a command:

      ```
      go
      ```

    - To clear the query buffer:

      ```
      reset
      ```

    - To execute an operating system command:

      ```
      !! command
      ```

    - To exit from isql:

      ```
      quit
      ```

      or

      ```
      exit
      ```

    - To redirect the output of a T-SQL command to a new file, or overwrite the file if it already exists:

      ```
      >
      ```

    - To redirect the output of a T-SQL command to a new file, or append to the file if it already exists:

      ```
      >>
      ```

    - To pipe the output of a T-SQL command to an external application from within an isql session:

      ```
      |
      ```

- isql is built with Client-Library. isql is built using the nonthreaded client libraries.
- isql_r is a threaded version of isql. You must use isql_r if a security service, such as Kerberos, or a directory service, such as LDAP, is used.
- Error message format differs from earlier versions of isql. If you have scripts that perform routines based on the values of these messages you may need to rewrite them.
- To use isql interactively, enter isql (and any of the optional flags) at your operating system prompt. The isql program accepts SQL commands and sends them to Adaptive Server. The results are formatted and printed on standard output. Exit isql with quit or exit.

- Terminate a command by typing a line beginning with the default command terminator go or other command terminator if the -c option is used. You may follow the command terminator with an integer to specify how many times to run the command. For example, to execute this command 100 times, type the following:

  ```
  select x = 1
  go 100
  ```

  The results appear once at the end of execution.

- If you enter an option more than once on the command line, isql uses the last value. For example, if you enter the following command, “send”, the second value for -c, overrides “.”, the first value:

  ```
  isql -c. -csend
  ```

  This enables you to override any aliases you set up.

- To call an editor on the current query buffer, enter its name as the first word on a line. Define your preferred callable editor by specifying it with the EDITOR environment variable. If EDITOR is undefined, the default is vi.

  For example, if the EDITOR environment variable is set to *emacs*, invoke it from isql using *emacs* as the first word on a line.

- Execute operating system commands by starting a line with two exclamation points (!!) followed by the command.

- To clear the existing query buffer, type reset on a line by itself. This entry uses isql to discard any pending point. You can also press Ctrl+C anywhere on a line to cancel the current query and return to the isql prompt.

- Read in an operating system file containing a query for execution by isql as follows:

  ```
  isql -U alma -P****** < input_file
  ```

  The file must include command terminators. The results appear on your terminal. Read in an operating system file containing a query and direct the results to another file as follows:

  ```
  isql -U alma -P****** < input_file > output_file
  ```

- isql flags are case-sensitive.

- isql displays only six digits of float or real data after the decimal point, rounding off the remainder.

- When using isql interactively, read an operating system file into the command buffer using:

```
:r filename
```

Do not include a command terminator in the file; enter the terminator interactively once you have finished editing.

- When using isql interactively, read and display an operating system file into the command buffer using:

```
:R filename
```

- When using isql interactively, you can change the current database using:

```
use databasename
```

- You can include comments in a Transact-SQL statement submitted to Adaptive Server by isql. Open a comment with "/*". Close it with "*/" as the following example demonstrates:

```
 select au_lname, au_fname
 /*retrieve authors' last and first names*/
 from authors, titles, titleauthor
 where authors.au_id = titleauthor.au_id
 and titles.title_id = titleauthor.title_id
 /*this is a three-way join that links authors
 **to the books they have written.*/
```

- If you want to comment out a go command, it should not be at the beginning of a line. For example, to comment out the go command, use:

```
/*
**go
*/
```

Do not use:

```
/*
go
*/
```

- isql defines the order of the date format as month, date, and year (mm dd yyyy hh:mm AM or PM), regardless of the locale environment. To change this default order, use the convert function.

Additional commands within isql:

Table A-7: isql session commands

Command	Description
>	Redirects command output to a file. File is overwritten if it exists.
>>	Redirects command output to a file. The output is appended to the file if the file already exists.
|	Pipes the output of a command to an external application.
reset	Clears the query buffer.
quit or exit	Exits from isql.
vi	Calls the editor.
!! *command*	Executes an operating system command.
:r *filename*	Reads an operating system file.
:R *filename*	Reads and displays an operating system file.
use *dbname*	Changes the current database to *dbname*.

Using prompt labels and double wildcards in an *isql* session

In an isql session, the default prompt label is either the default wildcard :? or the value of *wildcard*. You can customize the prompt label by providing a one-word character string with a maximum length of 80 characters, after a wildcard. If you specify a prompt label that is more than one word, the characters after the first word are ignored.

Double wildcards such as :?:? specify that isql needs to prompt you twice for the same input. The second prompt requests you to confirm your first input. If you use a double wildcard, the second prompt label starts with Confirm.

Note In an isql session, isql recognizes :? or the value of *wildcard* as wildcards only when you place these characters at the beginning of an isql line.

Using command history

- The command history feature is available only in command mode. Also, only commands that are issued interactively in isql are included in the command history. Examples of commands that are not included in the command history are those that are executed using the -i command line option or as part of a redirected input such as:

```
isql -Uguest -Ppassword -Smyase --history p1024
```

```
   --history_file myaseHistory.log <<EOF
exec sp_x_y_z
go
EOF
```

- Command history contains the most recent commands issued in an isql session. When *history_length* is reached, isql drops the oldest command from the history and adds the newest command issued.
- If you do not specify an alternate log file, and if the *$HOME* or *%APPDATA%* environment variable used by the default log file is not defined, an error message appears and the command history log is not saved.

In an isql session, use the h [*n*] command to display the command history. A page can display up to 24 lines of commands. If the command history contains more than 24 lines, press Enter to display the next set of commands or enter "a" to display all commands in one page. Enter "q" to return to isql.

n – indicates the number of commands to appear. If *n* is positive, the commands that appear start from the oldest command in the history. If *n* is negative, the *n* most recent commands appear.

Use the ? *n* | ?? command to recall and reissue a command from the command history.

n – when *n* is positive, isql looks for the command labeled with the number *n* and loads this to the command buffer. When *n* is negative, isql loads the *n*th most recent command issued.

?? – recalls the latest command issued and is equivalent to `? -1`.

- When a command is recalled from history, the recalled command overwrites the command in the command buffer.
- You can edit a recalled command before resubmitting the command to the server.

See also

sp_addlanguage, sp_addlogin, sp_configure, sp_defaultlanguage, sp_droplanguage, and sp_helplanguage in the *Adaptive Server Enterprise Reference Manual*.

APPENDIX B

Environment Variables

This appendix contains the values of the environment variables required for your Sybase applications to compile and work correctly. The environment variables that must be set depend on your application, and include:

- SYBASE – set to the path of the Sybase installation directory.
- SYBASE_OCS – set to the subdirectory containing the Open Client and Open Server version number. For example, *OCS-15_0* is the home directory of 15.5 version of the Open Client and Open Server products.
- DSQUERY – set to the name of the Adaptive Server or Open Server.
- DSLISTEN – set to the name of the Open Server.
- SYBPLATFORM – depends on the platform that you are running and whether or not you are using reentrant libraries. See Table B-1 for the appropriate variable setting.
- You must set the platform specific library path variable listed in Table B-1 to *$SYBASE/$SYBASE_OCS/lib* to run programs linked with shareable (dynamic) libraries. If you are running in debug mode, set the platform-specific library path variable to *$SYBASE/$SYBASE_OCS/devlib*.

  For ESQL/COBOL applications, include the location of the *$COBDIR/coblib* directory.

Table B-1: SYBPLATFORM and library path

Platform	SYBPLATFORM setting	Platform-specific library path variable
HP HP-UX PA-RISC 32-bit	hpux	SHLIB_PATH
HP HP-UX PA-RISC 32-bit using native threads	nthread_hpux	
HP HP-UX PA-RISC 64-bit	hpux64	LD_LIBRARY_PATH
HP HP-UX PA-RISC 64-bit using native threads	nthread_hpux64	
HP HP-UX Itanium 32-bit	hpia	SHLIB_PATH
HP HP-UX Itanium 32-bit using native threads	nthread_hpia	
HP HP-UX Itanium - 64-bit	hpia64	LD_LIBRARY_PATH
HP HP-UX Itanium - 64-bit using native threads	nthread_hpia64	
IBM AIX RS/6000 32-bit	rs6000	LIBPATH
IBM AIX RS/6000 32-bit using native threads	nthread_rs6000	
IBM AIX RS/6000 64-bit	rs600064	LIBPATH
IBM AIX RS/6000 64-bit using native threads	nthread_rs600064	
Linux x86 32-bit	linux	LD_LIBRARY_PATH
Linux x86 32-bit using native threads	nthread_linux	
Linux POWER 32-bit	ibmplinux	LD_LIBRARY_PATH
Linux POWER 32-bit using native threads	nthread_ibmplinux	
Linux POWER 64-bit	ibmplinux64	LD_LIBRARY_PATH
Linux POWER 64-bit using native threads	nthread_ibmplinux64	
Linux x86-64 64-bit	linux64 or linuxamd64	LD_LIBRARY_PATH
Linux x86-64 64-bit using native threads	nthread_linux64 or nthread_linuxamd64	
Solaris SPARC 32-bit	sun_svr4	LD_LIBRARY_PATH
Solaris SPARC 32-bit using native threads	nthread_sun_svr4	

Platform	SYBPLATFORM setting	Platform-specific library path variable
Solaris SPARC 64-bit	sun_svr464	LD_LIBRARY_PATH_64
Solaris SPARC 64-bit using native threads	nthread_sun_svr464	
Solaris x86-64 32-bit	sunx86	LD_LIBRARY_PATH
Solaris x86-64 32-bit using native threads	nthread_sunx86	
Solaris x86-64 64-bit	sunx8664	LD_LIBRARY_PATH_64
Solaris x86-64 64-bit using native threads	nthread_sunx8664	

For Embedded SQL/COBOL applications, set these environment variables in addition to the ones listed above:

- COBDIR – set to the path of your COBOL compiler.
- PATH – add $COBDIR/bin.

A P P E N D I X C

Utility Messages

This appendix describes error, warning, and information messages for the bcp, defncopy, and isql utilities.

- bcp messages
- defncopy messages
- isql messages

bcp messages

Message 1: Memory allocation failure

Message type	Error
Symbolic constant	BERRNOMEM
Message text	`Fatal error: memory allocation failed.`
Cause	The *start* argument is invalid.
Action	Make sure the *start* argument is smaller than the length of the language or RPC command.

Message 5: Unable to open input file

Message type	Error
Symbolic constant	BERRNOINFILE
Message text	`Unable to open input file '%1!'.`
Cause	bcp could not open the data file for input.
Action	Check the specified file.

Message 6: Unable to open output file

Message type	Error
Symbolic constant	BERRNOOUTFILE
Message text	`Unable to open output file '%1!'.`
Cause	bcp could not open the data file for output.
Action	Check the specified file.

Message 7: Bad arguments

Message type	Error
Symbolic constant	BERRBADARG
Message text	`bcp: Unknown parameter '%1!'.`
Cause	An unknown parameter was submitted.
Action	Correct the unknown parameter and resubmit.

Message 8: Invalid first row

Message type	Error
Symbolic constant	BERRFIRSTROW
Message text	`When using the '%1!' flag to set the first row to copy, the row number must be greater than or equal to 1.`
Cause	The specified first row is invalid.
Action	Correct the row number.

Message 9: Invalid rows

Message type	Error
Symbolic constant	BERRFLROW
Message text	`When using the '%1!' and '%2!' flags to set the first and last rows to copy, the first row number must be smaller than the last.`

Cause	The specified first or last row is invalid.
Action	Correct the row range so that the number of the first row is lower than the number of the last row.

Message 10: Invalid last row

Message type	Error
Symbolic constant	BERRLASTROW
Message text	`When using the '%1!' flag to set the last row to copy, the row number must be greater than or equal to 1.`
Cause	Correct the row number.
Action	Make sure the *start* argument is smaller than the length of the language or RPC command.

Message 11: Invalid direction

Message type	Error
Symbolic constant	BERRBADDIR
Message text	`Copy direction must be either 'in' or 'out'.`
Cause	The direction specified is invalid.
Action	Correct the direction.

Message 12: Invalid integer

Message type	Error
Symbolic constant	BERRBADINTARG
Message text	`The '%1!' flag must be followed by an integer.  '%2!' is not a legal integer.`
Cause	The argument specified is not an integer.
Action	Correct the argument.

Message 13: Duplicate flags

Message type	Warning
Symbolic constant	BERRDUPARGS
Message text	`Warning: the '%1!' flag appears more than once.  The new flag's value supersedes the old.`
Cause	Duplicate arguments have been specified.
Action	Remove one of the arguments.

Message 14: Overriding arguments

Message type	Warning
Symbolic constant	BERROVERRIDE
Message text	`Warning: '%1!' overrides '%2!'`
Cause	Two arguments override each other.
Action	Remove one of the arguments if necessary.

Message 15: Invalid prefix length

Message type	Error
Symbolic constant	BERRBADPREFXLEN
Message text	`Invalid prefix length. Valid prefix-lengths are 0, 1, 2, or 4.`
Cause	The prefix length is invalid.
Action	Provide a valid prefix length.

Message 21: Retry

Message type	Information
Symbolic constant	BSTRTRY
Message text	`Try again`
Cause	A non-fatal error occurred.

Action	Retry the operation.

Message 23: Starting message

Message type	Information
Symbolic constant	BSTRSTART
Message text	`Starting copy...`

Message 24: N rows copied

Message type	Information
Symbolic constant	BSTRROW
Message text	`%1! rows copied.`

Message 25: Total time

Message type	Information
Symbolic constant	BSTRTIME
Message text	`Clock Time (ms.): total = %1!`

Message 26: File save

Message type	Information
Symbolic constant	BSTRSAVE
Message text	`Do you want to save this format information in a file? [Y/n]`

Message 27: Host file

Message type	Information
Symbolic constant	BSTRHOST

Message text	`Host filename [%1!]:`

Message 28: Invalid column type

Message type	Error
Symbolic constant	BERRCOLTYPE
Message text	`Invalid column type. Valid types are:`
Cause	The column type specified is invalid.
Action	Provide a valid column type.

Message 29: Invalid column type

Message type	Information
Symbolic constant	BERRDSCOL
Message text	`<cr>: same type as DataServer column.`
Cause	The column type specified is invalid.
Action	Provide a valid column type.

Message 30: Average Time

Message type	Information
Symbolic constant	BSTRAVG
Message text	`Avg = %1! (%2! rows per sec.)`
	Indicates the average processing time per row.

Message 31: Copy failure

Message type	Error
Symbolic constant	BERRCOPY
Message text	`bcp copy %1! failed`
Cause	There was an error during the copy.

Action	Retry the copy operation.

Message 32: Partial copy failure

Message type	Error
Symbolic constant	BERRPCOPY
Message text	`bcp copy %1! partially failed`
Cause	Some rows were not copied.
Action	Retry the copy operation for the specified rows

Message 33: Invalid precision

Message type	Error
Symbolic constant	BERRBADPRECISION
Message text	`Invalid precision. Precision should be between %1! and %2!`
Cause	The precision specified is invalid.
Action	Provide a valid precision.

Message 34: Invalid scale

Message type	Error
Symbolic constant	BERRBADSCALE
Message text	`Invalid scale. Scale should be between %1! and %2! and should be less than or equal to the precision.`
Cause	The scale specified is invalid.
Action	Provide a valid scale.

Message 35: Unexpected result type

Message type	Error

Symbolic constant	BERRBADTYPE
Message text	`Unexpected result type returned.`
Cause	The server returned an incorrect result type.

Message 36: Unexpected result

Message type	Error
Symbolic constant	BERRBADRESULT
Message text	`Unexpected result returned.`
Cause	The server returned an incorrect result.

Message 37: Write error

Message type	Error
Symbolic constant	BERRWRITEERR
Message text	`Error: Writing BCP file (%1!)!`
Cause	An error occurred in writing to the bcp file.
Action	Check the specified file.

Message 39: Invalid rows

Message type	Error
Symbolic constant	BERRNOTENOUGHROWS
Message text	`The first row specified is greater than the no of rows in the table.`
Cause	The number of the first row specified is greater than the number of rows in the table.
Action	Provide a valid number for the first row.

Message 40: Row transfer error

Message type	Error
Symbolic constant	BERRXFERMULT
Message text	`blk_rowxfer_mult returned unexpected return code.`
Cause	The return code from the blk_rowxfer_mult routine is unexpected.

Message 41: Invalid datatype

Message type	Error
Symbolic constant	BERRDATATYPE
Message text	`Unknown data type '%1!' encountered.`
Cause	An invalid datatype was encountered during the copy.

Message 42: Input read file error

Message type	Error
Symbolic constant	BERRIOERR
Message text	`I/O error while reading the bcp input-file.`
Cause	An error occurred in reading the bcp input file.
Action	Check the specified file.

Message 43: Error file write error

Message type	Error
Symbolic constant	BERRWEF
Message text	`I/O error while writing bcp error-file.`
Cause	An error occurred in writing to the bcp error file.
Action	Check the specified file.

Message 44: Unable to open error file

Message type	Error
Symbolic constant	BERRUOE
Message text	`Unable to open error-file.`
Cause	bcp could not open the error file.
Action	Check the specified file.

Message 45: Unexpected end-of-file

Message type	Error
Symbolic constant	BERREOF
Message text	`Unexpected EOF encountered in BCP data-file.`
Cause	There was an unexpected end-of-file character in the bcp data file.
Action	Check the specified file content.

Message 46: Negative-length prefix

Message type	Error
Symbolic constant	BERRCNL
Message text	`Negative length-prefix found in BCP data-file.`
Cause	A negative-length prefix was found in the bcp data file.
Action	Provide a valid length prefix in the bcp data file.

Message 48: Cannot read specified number of rows

Message type	Error
Symbolic constant	BERRSHORTFILE
Message text	`The BCP hostfile '%1!' contains only %2! rows. It was impossible to read the requested number of rows.`
Cause	There are fewer rows in the bcp host file than were requested to read.

Action	Specify a number of rows to read that is less than or equal to the number of rows in the bcp host file.

Message 49: Length prefix or terminator required

Message type	Error
Symbolic constant	BERRVDPT
Message text	`For bulk copy, all variable-length data must have either a length-prefix or a terminator specified.`
Cause	For bulk copy, all variable-length data must have either a length prefix or a terminator specified.
Action	Provide a length prefix or terminator.

Message 50: Text/image data truncated

Message type	Error
Symbolic constant	BERRTRUNDATA
Message text	`Text/image field is larger than the maximum value. Data truncated.`
Cause	The text or image data is larger than the maximum size. Any data beyond the maximum has been truncated.

Message 51: Max errors exceeded

Message type	Error
Symbolic constant	BERRMAXERROR
Message text	`The total number of errors in this BCP operation is greater than the maximum number of errors (%1!) allowed. BCP has stopped.`
Cause	The bcp operation exceeded the maximum total number of errors allowed.

Message 52: Unable to open discard file

Message type	Error
Symbolic constant	BERRUOD
Message text	`Unable to open the discard-file '%1!'.`
Cause	bcp could not open the discard file.
Action	Check the specified file.

Message 53: Discard file write error

Message type	Error
Symbolic constant	BERRWDF
Message text	`I/O error while writing the bcp discard-file '%1!'.`
Cause	An error occurred in writing to the bcp discard file.
Action	Check the specified file.

Message 54: Unable to close file

Message type	Error
Symbolic constant	BERRECF
Message text	`Unable to close the file '%1!'. Data may not have been copied.`
Cause	An error occurred in closing the file.
Action	Check the specified file.

Message 55: Batch size adjusted

Message type	Warning
Symbolic constant	BERRDISCBATWAR
Message text	`Warning: Batch size adjusted to the value '%1!', for the optimization of the discard-file feature.`

Cause	The maximum memory usage has been exceeded, and the array size has been reduced.

Message 56: Max rows reached

Message type	Error
Symbolic constant	BERRMAXROWNUM
Message text	`The maximum row number that bcp can process is reached, total number of '%1!' rows have been processed, bcp operation terminated.`
Cause	The bcp operation has processed the maximum number of rows and has been terminated.

defncopy messages

Message 1: Memory allocation failure

Message type	Error
Symbolic constant	ERRNOMEM
Message text	`Fatal error: memory allocation failed.`
Cause	The Client-Library application is unable to allocate memory.
Action	Check the memory available to your application. Increase physical or virtual memory, or terminate other applications to free memory.

Message 2: Insufficient read space

Message type	Error
Symbolic constant	ERRNOREADSPACE
Message text	`I/O Error: Insufficient space for input data.`
Cause	There is insufficient space in the read buffer.

Action	Check the input buffer.

Message 3: Unable to open input file

Message type	Error
Symbolic constant	ERRNOINFILE
Message text	`Unable to open input file '%1!'.`
Cause	defncopy could not open the data file for input.
Action	Check the specified file.

Message 4: Unable to open output file

Message type	Error
Symbolic constant	ERRNOOUTFILE
Message text	`Unable to open output file '%1!'.`
Cause	defncopy could not open the data file for output.
Action	Check the specified file.

Message 5: Bad argument

Message type	Error
Symbolic constant	ERRBADARG
Message text	`defncopy: Unknown parameter '%1!'.`
Cause	An unknown parameter was submitted.
Action	Provide a valid parameter.

Message 6: File not flushed

Message type	Error
Symbolic constant	ERRNOFLUSH

Message text	`Failed to flush file '%1!': '%2!'.`
Cause	The operating system failed to flush the specified file.

Message 7: Unexpected object definition

Message type	Error
Symbolic constant	ERRNOOBJDEF
Message text	`Definition of object '%1!' not found.`

Message 8: Abend

Message type	Error
Symbolic constant	ERRABORT
Message text	`defncopy aborted.`
Cause	The interrupt handler was triggered.

Message 9: Invalid direction

Message type	Error
Symbolic constant	ERRBADDIRECTION
Message text	`(direction must be either 'in' or 'out'.)`
Cause	The direction specified is invalid.
Action	Correct the direction.

Message 10: No object name

Message type	Error
Symbolic constant	ERRNOOBJNAME
Message text	`(at least one object name needed.)`
Cause	No object name was provided.
Action	Provide at least one object name.

isql messages

Message 1: Memory allocation failure

Message type	Error
Symbolic constant	LOC_ENBR_ERR_MALLOC
Message text	`Fatal error: memory allocation failed.`
Cause	The Client-Library application is unable to allocate memory.
Action	Check the memory available to your application. Increase physical or virtual memory, or terminate other applications to free memory.

Message 8: Database name length

Message type	Error
Symbolic constant	LOC_ENBR_ERR_LONGDBNAME
Message text	`Database name too long.`
Cause	The specified database name is too long.
Action	Provide a database name of valid length.

Message 9: CS-Lib message callback routine installation

Message type	Error
Symbolic constant	LOC_ENBR_ERR_CSMSGCB
Message text	`Unable to install CS-Library message callback routine.`
Cause	isql could not install an error handler.

Message 10: CT-Lib initialization

Message type	Error
Symbolic constant	LOC_ENBR_ERR_INITCTLIB
Message text	`Unable to initialize Client Library.`

Cause	A call to the ct_init function failed.
Action	Restart the application.

Message 11: CT-Lib message callback routine installation

Message type	Error
Symbolic constant	LOC_ENBR_ERR_CTMSGCB
Message text	`Unable to install Client Library client message callback routine.`
Cause	A call to the ct_callback function failed.
Action	Restart the application.

Message 12: Unsupported datatype

Message type	Error
Symbolic constant	LOC_ENBR_ERR_DATATYPE
Message text	`Unsupported datatype encountered.`
Cause	An invalid datatype was specified.
Action	Correct the invalid argument.

Message 13: Buffer overflow

Message type	Error
Symbolic constant	LOC_ENBR_ERR_BUFFOVFLW
Message text	`Buffer overflow occurred while printing row.`
Cause	There is too much data to print.

Message 15: Invalid memory block size

Message type	Error
Symbolic constant	LOC_ENBR_ERR_INVMEMBLCK

Message text	`Invalid memory block size specified.`
Cause	The Client-Library application is unable to allocate memory.
Action	Check the memory available to your application.

Message 16: Invalid memory handle

Message type	Error
Symbolic constant	LOC_ENBR_ERR_INVMEMHNDL
Message text	`Invalid memory handle specified.`
Cause	The Client-Library application is unable to allocate memory.
Action	Check the memory available to your application.

Message 17: Internal memory allocation error

Message type	Error
Symbolic constant	LOC_ENBR_ERR_INTMALLOC
Message text	`Internal isql memory allocation error.`
Cause	The Client-Library application is unable to allocate memory.
Action	Check the memory available to your application.

Message 18: Editor command too long

Message type	Error
Symbolic constant	LOC_ENBR_ERR_LONGEDCMDLN
Message text	`Command line to invoke editor too long.`
Cause	The command invoked to start the editor is too long.
Action	Reduce the command to a valid length.

Message 19: Uninitialized application context

Message type	Error
Symbolic constant	LOC_ENBR_ERR_APPCONTEXT
Message text	`An isql application context has not been initialized.`
Cause	A call to the ct_config function failed.
Action	Restart the application.

Message 20: Connection failure

Message type	Error
Symbolic constant	LOC_ENBR_ERR_CONFAILED
Message text	`A connection with a server has not been established.`
Cause	A call to the ct_connect function failed.
Action	Reattempt to connect to the server.

Message 21: Unavailable command handle

Message type	Error
Symbolic constant	LOC_ENBR_ERR_NOCMDHNDL
Message text	`No command handle is available.`
Cause	There is no command handle.
Action	Restart the application.

Message 23: File position reset failure

Message type	Error
Symbolic constant	LOC_ENBR_ERR_RSTPOSIND
Message text	`Failed to reset the file's position indicator to the beginning of the file.`
Cause	The operating system failed to reposition the indicator.

Action	Restart the application.

Message 24: Command buffer not cleared

Message type	Error
Symbolic constant	LOC_ENBR_ERR_CLRCMDBUF
Message text	`Failed to clear the command buffer.`
Cause	A call to the ct_cancel function failed.
Action	Restart the application.

Message 25: Command not initiated

Message type	Error
Symbolic constant	LOC_ENBR_ERR_INITCMD
Message text	`Unable to initiate the command.`
Cause	A call to the ct_command function failed.
Action	Restart the application.

Message 26: Command handle not cleared

Message type	Error
Symbolic constant	LOC_ENBR_ERR_CLRCMDHNDL
Message text	`Failed to clear the command in the command handle.`
Cause	A call to the ct_cancel function failed.
Action	Restart the application.

Message 28: Command argument too long

Message type	Error
Symbolic constant	LOC_ENBR_ERR_LONGCMDLN

Message text	`Command line too long.`
Cause	A command argument is too long.
Action	Provide an argument of valid length.

Message 29: Filename missing

Message type	Error
Symbolic constant	LOC_ENBR_ERR_FILENAME
Message text	`Missing file or executable name.`
Cause	A file name or executable name is missing.
Action	Provide a name for the file or executable.

Message 30: Prompt label too long

Message type	Error
Symbolic constant	LOC_ENBR_ERR_LONGPRMPTLBL
Message text	`The prompt label is too long.`
Cause	The prompt label is too long.
Action	Provide a prompt label of valid length.

Message 31: Prompt input mismatch

Message type	Error
Symbolic constant	LOC_ENBR_ERR_DIFFINPUT
Message text	`Input from 1st prompt is different from input from confirmation prompt.`
Cause	Input from the first and confirmation prompts does not match.
Action	Provide the same input for both prompts.

Message 32: Missing quote

Message type	Error
Symbolic constant	LOC_ENBR_ERR_QUOTES
Message text	`The quoted file name is missing the closing quote.`
Cause	The file name has a starting quote but no ending quote.
Action	Add the missing quote.

Message 33: Directory creation failure

Message type	Error
Symbolic constant	LOC_ENBR_ERR_PRIVDIR
Message text	`Failed to create directory '%1!': '%2!'`
Cause	isql could not create the specified directory.
Action	Check the directory containing the isql command history.

Message 34: Unexpected argument type

Message type	Error
Symbolic constant	LOC_ENBR_ERR_PRIVDIRTYPE
Message text	`Found unexpected type for '%1!': '%2!'`
Cause	isql could not determine the directory type for the command history file.
Action	Check the directory containing the isql command history.

Message 35: Unable to open history file

Message type	Error
Symbolic constant	LOC_ENBR_ERR_LOGWRITE
Message text	`Failed to open for write '%1!': '%2!'`
Cause	isql could not open the command history file for writing.
Action	Check the directory containing the isql command history.

Message 36: Temporary file deletion failure

Message type	Error
Symbolic constant	LOC_ENBR_ERR_TMPFILEDEL
Message text	`Failed to delete temporary file '%1!'`
Cause	isql could not delete the specified temporary file.

Index