SYBASE®

Programmers Supplement

Open Client™ and Open Server™

15.0

[UNIX]

DOCUMENT ID: DC35456-01-1500-07

LAST REVISED: December 2008

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.

The following UNIX platforms are covered:

- HP 9000/800 (HP-UX 11.11 or later) 32-bit and 64-bit
- HP Itanium (HP-UX 11.23 or later) 32-bit and 64-bit
- IBM RS/6000 (AIX 5.2 or later) 32-bit and 64-bit
- Linux x86 32-bit
- Linux on POWER 32-bit and 64-bit
- Linux AMD64 (Opteron)/EM64T
- Sun Solaris 2.8 (SPARC) or later 32-bit and 64-bit
- Sun Solaris 10 x64 (Opteron) 32-bit and 64-bit

Audience

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

How to use this book

This supplement contains material for all the Open Client and Open Server products running on the UNIX operating system. Each product, such as Open Client Client-Library or Open Server, is covered in its own chapter. The chapters discuss issues such as:

- Building an executable
- Information on the sample programs, including their locations and what they do

The appendixes contain the following:

- Reference pages that detail the syntax, parameters, and qualifiers for the commands and utilities relevant to Open Client and Open Server
- Information about the environment variables you need to set so that you can build and run your applications

Related documents

Each Open Client and Open Server product has its own set of user documentation. Table 1 lists the products and their related documents:

Table 1: Product documentation list

Product	Related Documentation
Client-Library	Open Client *Client-Library/C Reference Manual* Open Client and Open Server *Common Libraries Reference Manual* Open Client *Client-Library/C Programmers Guide*
DB-Library	Open Client *DB-Library/C Reference Manual*
Server-Library	Open Server *Server-Library/C Reference Manual* Open Client and Open Server *Common Libraries Reference Manual*
ESQL/C	Open Client *Embedded SQL/C Programmers Guide*
ESQL/COBOL	Open Client *Embedded SQL/COBOL Programmers Guide*

See your installation guide for information on installation, directory structure, and logical names.

See the Open Client and Open Server *Configuration Guide* for UNIX for information on how to:

- Set up your environment so that Open Client applications and servers can communicate
- Localize Sybase applications

See the Open Server™ and SDK *New Features* for Microsoft Windows, Linux, UNIX, and Mac OS X for descriptions of new features available for Open Server and the Software Developer's Kit. This document is revised to include new features as they become available.

Other sources of information

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

- The Getting Started CD contains release bulletins and installation guides in PDF format, and may also contain other documents or updated information not included on the SyBooks CD. 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 SyBooks CD contains product manuals and is included with your software. The Eclipse-based SyBooks browser allows you to access the manuals in an easy-to-use, HTML-based format.

  Some documentation may be provided in PDF format, which you can access through the PDF directory on the SyBooks CD. To read or print the PDF files, you need Adobe Acrobat Reader.

  Refer to the *SyBooks Installation Guide* on the Getting Started CD, or the *README.txt* file on the SyBooks CD for instructions on installing and starting SyBooks.
- The Sybase Product Manuals Web site is an online version of the SyBooks CD that you can access using a standard Web browser. In addition to product manuals, you will find links to EBFs/Maintenance, Technical Documents, Case Management, Solved Cases, newsgroups, and the Sybase Developer Network.

  To access the Sybase Product Manuals Web site, go to Product Manuals 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 Certification Report.
3. In the Certification Report filter select a product, platform, and timeframe and then click Go.
4. Click a 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 2 describes the syntax conventions used in this manual:

Table 2: 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*.

Key	Definition
{ }	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 manuals 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 for use in writing 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 which is allocated in CS-Library.

This chapter covers the following topics:

Topic	Page
General instructions	1
Building a Client-Library executable	2
Using Client-Library sample programs	12

Note Refer to 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.

General instructions

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

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

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

Building a Client-Library executable

This section discusses the libraries and compile and link lines needed 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 non-threaded 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

Client-Library version includes thread-safe libraries which allows developers to create multithreaded applications using POSIX threads.

Refer to “Compile-and-link lines for multithreaded applications” on page 8 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)
Sun 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 9000 (8xx) and HP Itanium	*libcl* – HP Transport Control Layer (system) *libBSD* – The BSD library (system) *libc_r* – C reentrant library *libdld* – (system)
IBM RS/6000	*libc_r* – C reentrant library *libpthreads* – Thread library (system)
Linux platforms	*libpthread* – Thread library (system)

Kerberos support

Client-Library version 11.1 and later supports Kerberos security features for applications that desire 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 that are supported in this version:

- Network authentication
- Mutual authentication
- Out-of-sequence authentication
- Replay detection
- Confidentiality
- Integrity
- Credential delegation (available in SDK 15.0 ESD #10 and later)

To develop and run Client-Library applications that take advantage of Kerberos features, perform the tasks listed in Table 1-3.

Table 1-3: Required tasks for Kerberos support

Tasks	For more information
Install the Kerberos software on your system.	Refer to your Kerberos documentation and the Open Client and Open Server *Configuration Guide* for UNIX for instructions.
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.	Refer to your Kerberos documentation.
Set the environment variable to the credential cache directory location.: • For CyberSafe, CSFC5CCNAME • For MIT, KRB5CCNAME	Refer to 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. This means that you must 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, refer to the *Makefile* and *sybopts.sh* file in the *$SYBASE/$SYBASE_OCS/sample/ctlibrary* directory 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
Sun Solaris 2.x	`/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`
Sun Solaris 10 x64 (Opteron) 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 RS/6000	`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 9000 (8xx)	`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 Itanium	`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	`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 on POWER	`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 AMD64 (Opteron)/ EM64T	`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
Sun Solaris 2.x	`/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`
Sun Solaris 10 x64 (Opteron) 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 RS/6000	`xlc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lm -o program`
HP 9000 (8xx)	`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 Itanium	`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	`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 on POWER	`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 AMD64 (Opteron)/ EM64T	`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
Sun Solaris 2.8	`/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`
Sun Solaris 10 x64 (Opteron) 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 RS/6000	`xlc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bdynamic -lsybct -lsybcs -lm -o program`
HP 9000 (8xx)	`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 Itanium	`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	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bdynamic -lsybct -lsybcs -ldl -lnsl -lm -o program`
Linux on POWER	`xlc -q32 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bdynamic -lsybct -lsybcs -ldl -lnsl -lm -o program`
Linux AMD64 (Opteron)/ EM64T	`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
Sun Solaris 2.8	`/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`
Sun Solaris 10 x64 (Opteron) 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 RS/6000	`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 9000 (8xx)	`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 Itanium	`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	`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 on POWER	`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 AMD64 (Opteron)/ EM64T	`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-UX and HP Itanium 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-UX and HP Itanium will 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 options so that the system will be able to 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 for more information.
- 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.

  The environment variable LD_LIBRARY_PATH is platform-specific. See Table 1-8 for a list of environment variables to specify for each platform.

Table 1-8: LD_LIBRARY_PATH for UNIX platforms

Platform	Environment Variable
Sun Solaris 32-bit, Sun Solaris 10 x64 (Opteron) 32-bit, HP-UX 64-bit, HP Itanium 64-bit, Linux 32-bit, Linux on POWER 32-bit and 64-bit, Linux AMD64 (Opteron)/EM64T	LD_LIBRARY_PATH
Sun Solaris 64-bit, Sun Solaris 10 x64 (Opteron) 64-bit	LD_LIBRARY_PATH_64
HP-UX 32-bit, HP Itanium 32-bit	SHLIB_PATH
IBM AIX 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. This needs to be in the search path that the dynamic loader uses. Due to constraints in the implementation of the Sybase driver, only re-entrant libraries are supported when using Kerberos.

Bulk copy routines

If you plan to use bulk copy routines, link in the *libsybblk* library. If you plan to use bulk-copy routines in a threaded applications, link in the *libsybblk_r* library.

To link in the bulk-copy library:

- In non-threaded 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* for more information on bulk copying.

Performance considerations

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 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* for more information about header files.

Using Client-Library sample programs

Sample programs are included with Client-Library to demonstrate typical uses for Client-Library routines. Use the following information to compile and run the samples.

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

Makefile and sample programs

In order to use the *makefile* to build sample programs on all platforms, you must set the SYBPLATFORM environment variable correctly for the compiler you are using. For the environment variables and library path refer to Table B-1 on page 166.

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* script is included with the sample programs and helps you build Open Client and Open Server applications for your platform by reading the SYBPLATFORM environment variable:

```
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 available in *$SYBASE/$SYBASE_OCS/sample/ctlibrary* directory.

Location

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

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 file, *example.h.*
- The *README* file containing instructions for building, executing, and testing the samples.

Note 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_VERSION_150
#define EX_BLK_VERSION      BLK_VERSION_150
#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");
        }
    }
```

Values for these lines in *example.h* are described in the following sections.

EX_USERNAME

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

EX_PASSWORD

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

You have three options regarding EX_PASSWORD. Choose the one that best meets your needs:

- *Option 1* – Change your server password to 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.

- *Option 2* – 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.”
- *Option 3* – In the sample programs, modify the ct_con_props code that sets the server password and substitute your own code to prompt samples users 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 of the 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 generic routines that are used by the wide_* sample programs. The routines are as follows:

- 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

The following sample programs are included with your software.

arraybind.c sample program

The *arraybind.c* sample program demonstrates the use of array binding in conjunction 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.

For more information about this sample program, see the leading comments in the example source file.

Note This sample requires the pubs2 database.

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. For more information about this sample program, see the leading comments in the sample source file.

compute.c sample program

The *compute.c* sample program demonstrates processing of compute results and performs the following:

- It sends a canned query to the server using a language command.
- 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.

Following is the canned 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 compute rows. The compute 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)
  ```

For more information about this sample program, see the leading comments in the sample source file.

Note This sample requires the pubs2 database.

csr_disp.c sample program

The *csr_disp.c* sample program demonstrates using a read-only cursor:

- It opens a cursor with a canned 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.

Following is the canned query:

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

For more information about this sample program, see the leading comments in the example source file.

Note This sample requires a version 10.0 or later, and the pubs2 database.

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. It performs the following:

- It sends a canned query to the server to open a cursor.
- 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_scroll_fetch while loop.

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

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

For more information about this sample program, see the leading comments in the example source file.

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

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. It performs the following:

- It sends a canned query to the server to open a cursor.
- It processes the results using the standard ct_results while loop.
- It binds the column values to program variables.
- It fetches the rows using ct_scroll_fetch and displays them.

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 canned query:

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

For more information about this sample program, see the leading comments in the example source file.

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

csr_disp_implicit.c sample program

The *csr_disp_implicit.c* sample program demonstrates using an implicit read-only cursor:

- It opens a cursor with a canned 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.

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, it is important to set the CS_CURSOR_ROWS option to a value greater than 1.

This is the canned query:

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

For more information about this sample program, see the leading comments in the example source file.

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

ex_alib.c and *ex_amain.c* sample programs

This sample program contains two files, *ex_alib.c* and *ex_amain.c*, which demonstrate 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 Client-Library's 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 which supports asynchronous calls. This module provides a means of sending a query to and retrieving the results from a server within one asynchronous operation.
- *ex_amain.c* contains the source code to the main program that uses the services provided by *ex_alib.c*.

For more information about this sample program, see the leading comments in the example source file and the *EX_AREAD.ME* file.

exconfig.c sample program

The *exconfig.c* sample program demonstrates how Client-Library application properties can be configured externally.

This sample requires you to edit the default runtime configuration file, *$SYBASE/$SYBASE_OCS/config/ocs.cfg* file. 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 the topics page “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 text along with other datatypes. The process demonstrated can also be used for retrieving and updating image data. For more information about this sample program, see the leading comments in the example source file.

Note This example requires Adaptive Server version 10.0 or later.

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

For more information about this program, see the leading comments in the example source file.

multthrd.c and *thrdfunc.c* sample programs

This sample program contains two files, *multthrd.c* and *thrdfunc.c*, which demonstrate a multithreaded Client-Library application. The following information is contained in the two files:

- *multthrd.c* contains the 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 example uses for execution depending on your platform.

For more information about this program, see the leading comments in the example source file.

This sample cannot run if the 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. For more information about this sample program, see the leading comments in the example source file.

Note This example requires a Adaptive Server version 4.9.1 or later.

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 ASE or the *secsrv.c* Open Server sample program.

For more information about this sample program, see the leading comments in the example source file. For more information about network security services, refer to 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 to copy static data to a server table. This program has been modified for the use of the unichar and univarchar datatypes. 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.

uni_compute.c sample program

The *uni_compute.c* sample program demonstrates processing of 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 canned query to the server using a language command. After it processes the results using the standard ct_results loop, it binds the column values to program variables. Then, it fetches and displays the rows in the standard ct_fetch loop.

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

uni_csr_disp.c sample program

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

- It opens a cursor with a canned 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.

Following is the canned query:

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

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

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 available in the *$SYBASE/$SYBASE_OCS/sample/ctlibrary* directory.

For more information about this program, see the leading comments in the example source file.

usedir.c sample program

The *usedir.c* sample program demonstrates Client-Library's ability 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 being 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 this program, see the leading comments in the example source file. For more information about network directory services, refer to the Open Client and Open Server *Configuration Guide* for UNIX.

wide_compute.c sample program

The *wide_compute.c* sample program demonstrates processing of compute results with wide tables and larger column sizes. It performs the following:

- It sends a canned query to the server using a language command.
- 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.

Following is the canned 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 compute rows. The compute 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)
  ```

For more information about this sample program, see the leading comments in the sample source file.

Note This sample requires the pubs2 database.

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, you only pass 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.

For more information about this program, see the leading comments in the example source file.

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. New functionality such as Directory and Security services support is not included with DB-Library. You must use Client-Library to take advantage of these new 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.

This chapter covers the following topics:

General instructions

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

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

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

See the following sections for descriptions and additional requirements of the individual sample programs.

Building a DB-Library executable

This section gives information on libraries, linking, and header files.

Libraries

Include the libraries for all platforms if you want 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
Sun Solaris 2.x	`/opt/SUNWspro/bin/cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Bstatic -lsybdb -lsybunic -o program`
Sun Solaris 10 x64 (Opteron) 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 RS/6000	`xlc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bstatic -lsybdb -lsybunic -o program`
HP 9000 (8xx)	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-a,archive -lsybdb -lsybunic -Wl,-E,+s -o program`
HP Itanium	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-a,archive -lsybdb -lsybunic -Wl,-E,+s -o program`
Linux	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bstatic -lsybdb -lsybunic -ldl -o program`
Linux on POWER	`xlc -q32 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bstatic -lsybdb -lsybunic -ldl -o program`
Linux AMD64 (Opteron)/ EM64T	`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
Sun Solaris 2.x	`/opt/SUNWspro/bin/cc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybdb -lsybunic -o program`
Sun Solaris 10 x64 (Opteron) 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 RS/6000	`xlc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybdb -lsybunic -o program`
HP 9000 (8xx)	`cc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybdb -lsybunic -Wl,-E,+s -o program`
HP Itanium	`cc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybdb -lsybunic -Wl,-E,+s -o program`
Linux	`cc -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybdb -lsybunic -ldl -o program`
Linux on POWER	`xlc -q32 -g -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/devlib program.c -lsybdb -lsybunic -ldl -o program`
Linux AMD64 (Opteron)/ EM64T	`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
Sun Solaris 2.x	`/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`
Sun Solaris 10 x64 (Opteron) 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 RS/6000	`xlc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bdynamic -lsybdb -o program`
HP 9000 (8xx)	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,a,shared_archive -lsybdb -o program`
HP Itanium	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,a,shared_archive -lsybdb -o program`
Linux	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bdynamic -lsybdb -ldl -o program`
Linux on POWER	`xlc -q32 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bdynamic -lsybdb -ldl -o program`
Linux AMD64 (Opteron)/ EM64T	`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 linked with shared libraries may be slower at start-up time than those linked with static libraries. Also, unlike static libraries, the shared libraries must be available at runtime.

The individual requirements of your site determine which type of library will provide the best performance.

Header files

The following header files are required by all DB-Library/C applications:

- *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* for more information on header files.

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. Refer to 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* directory.

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

Note 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);
```

Following are descriptions of the changes you can make for the lines in *sybdbex.h*:

- USER is defined in *sybdbex.h* as “sa.” Before running the sample programs, you must edit *sybdbex.h* and change “sa” to your server login name.
- PASSWORD is defined in *sybdbex.h* as null (“ ”) string. Before running the sample programs, edit *sybdbex.h* and change “server_password” to your server password. Choose one of the following options for PASSWORD:

  Option 1: 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.

  Option 2: 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.”

  Option 3: 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.)

- LANGUAGE: 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

The following sample programs are included with your software.

Example 1: Send queries, bind, and print results

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

Example 2: Insert data into a new table

The *example2.c* sample program 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.

Example 3: Bind aggregate and compute results

The *example3.c* sample program 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 Access to Adaptive Server and the pubs2 database is required.

Example 4: Row buffering

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

Example 5: Data conversion

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

Example 6: Browse mode updates

The *example6.c* sample program 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.

Example 7: Browse mode and ad hoc queries

The *example7.c* sample program 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 only update 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. Notice how the results differ depending on whether the select query includes the keywords for browse and whether the table selected is able to be browsed.

Example 8: Making a remote procedure call (RPC)

The *example8.c* sample program 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.

Example 9: Text and image routines

The *example9.c* sample program generates a random image, inserts it into a table, then selects the image and compares it to the original by following these steps:

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 will have a valid text pointer only if the null value was explicitly entered with the update statement.
3. select the row. You must specifically select the column that is to contain the text or image value. This step is necessary to provide the application's DBPROCESS with correct text pointer and text timestamp information. The application should throw away the data returned by this select.
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 stored for later retrieval using dbtxtsput.

Note Access to an Adaptive Server that contains the pubs2 database is required.

Example 10: Inserting an image

The *example10.c* sample program 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 Example 9.

Note Access to an Adaptive Server that contains the pubs2 database is required. The author ID must be in the form "000-00-0000." The *imagefile* file, provided with the sample code, contains an image.

Example 11: Retrieving an image

The *example11.c* sample program 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 in conjunction 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 Access to Adaptive Server and the pubs2 database is required.

Example 12: International language routines

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

Note Access to Adaptive Server and the pubs2 database is required.

Example 13: Bulk copy

The bulk-copy sample program, *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 Access to Adaptive Server is required. You must have create database and create table permission.

Example 14: Two-phase commit

The two-phase commit sample program, *twophase.c,* 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, you can use isql on each of the servers 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, you need to make sure that your client can access both servers. Refer to the Open Client and Open Server *Configuration Guide* for UNIX for information about connecting to multiple instances of Adaptive Server.

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. For details, see the Open Client and Open Server *Configuration Guide* for UNIX.

CHAPTER 3

Open Server Server-Library/C

Open Server Server-Library/C is used to design servers that take advantage of the features of the client/server architecture. These Open Servers access data stored in foreign 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.

This chapter covers the following topics:

General instructions

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

- Be able to access Adaptive Server and the `pubs2` sample database. Refer to the Adaptive Server Enterprise *Installation Guide* for information on installing the `pubs2` database.
- Set the following 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. Refer to the Open Client and Open Server *Configuration Guide* for UNIX for information about connecting to an Adaptive Server.
- Set execute permission on the *sybopts.sh* file for the file's owner:

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

Building a Server-Library executable

This section gives information on libraries, linking, and header files.

Libraries

Table 3-1 lists the libraries that you should include if you want 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)
Sun 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 9000 (8xx) and HP Itanium	*libcl* – HP Transport Control Layer (system) *libBSD* – The BSD library (system) *libc_r* – C reentrant library *libdld* – (system)
IBM RS/6000	*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. The three tables include compile-and-link line commands.

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
Sun Solaris 2.x	`/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`
Sun Solaris 10 x64 (Opteron) 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 RS/6000	`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 9000 (8xx)	`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 Itanium	`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	`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 on POWER	`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`

Platform	Command
Linux AMD64 (Opteron)/ EM64T	`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
Sun Solaris 2.x	`/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`
Sun Solaris 10 x64 (Opteron) 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 RS/6000	`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 9000 (8xx)	`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 Itanium	`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	`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 on POWER	`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 AMD64 (Opteron)/EM64T	`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
Sun Solaris 2.x	`/opt/SUNWspro/bin/cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -R$SYBASE/lib program.c -Bdynamic -lsybsrv [ -lsybdb | -lsybct ] -lsybcs -lnsl -ldl -lm -lsocket -o program`
Sun Solaris 10 x64 (Opteron) 64-bit	`/opt/SunStudio10/SUNWspro/bin/cc -xtarget=opteron -xarch=amd64 -I$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 RS/6000	`xlc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib -R$SYBASE/lib program.c -Wl,-Bdynamic -lsybsrv [-lsybdb| -lsybct] -lsybcs -lm -o program`
HP 9000 (8xx)	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,a,shared_archive -lsybsrv [ -lsybdb | -lsybct ] -lsybcs -lcl -lm -lBSD -o program`
HP Itanium	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,a,shared_archive -lsybsrv [ -lsybdb | -lsybct ] -lsybcs -lcl -lm -lBSD -o program`
Linux	`cc -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bdynamic -lsybsrv [ -lsybdb | -lsybct ] -lsybcs -ldl -lnsl -lm -o program`
Linux on POWER	`xlc -q32 -I$SYBASE/$SYBASE_OCS/include -L$SYBASE/$SYBASE_OCS/lib program.c -Wl,-Bdynamic -lsybsrv [ -lsybdb | -lsybct ] -lsybcs -ldl -lnsl -lm -o program`
Linux AMD64 (Opteron)/ EM64T	`gcc -I$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
Sun Solaris 2.8	`/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`
Sun Solaris 10 x64 (Optero)n 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 RS/6000	`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 9000 (8xx)	`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 Itanium	`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	`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 on POWER	`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 AMD64 (Opteron)/ EM64T	`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 versions 11.1 and later support 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 the following Kerberos security features that are supported in this version:

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

To develop and run Server-Library applications that take advantage of Kerberos features, perform the tasks listed in Table 3-6:

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.	Refer to your Kerberos documentation and to 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.	Refer to 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 re-entrant libraries.	See "Kerberos support" on page 50.
• 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.	Refer to 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 each Open Server be run 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* for more information on bulk copying.

Performance considerations

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 individual requirements of your site determine which type of library will provide the best performance.

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 on how to configure directory services, including the *interfaces* file.

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 prior to attempting to use the sample programs.

Note These simplified programs are not intended for use in a production environment. Production-quality programs require additional error-handling and special-case-handling.

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

Location

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

- Source code for the sample programs.
- The *makefile* provided to build the samples. Use them 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

The following sample programs are included with your software.

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

ctos.c sample program

The *ctos.c* sample program 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. This program *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, it 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*.

exfds.c sample program

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

- 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 is used to allow 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 and 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* sample program is an Open Server gateway application that uses the Sybase Tabular Data Stream™ (TDS) passthrough mode. For more information on TDS passthrough, see the “Passthrough Mode” topics page 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.

intlchar.c sample program

The *intlchar.c* sample program demonstrates Open Server's 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* sample program demonstrates the use of 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 the “Language Calls” topics page in the Open Server *Server-Library/C Reference Manual*.

multthrd.c sample program

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

regproc.c sample program

The *regproc.c* sample program demonstrates the use of registered procedures in Open Server versions 11.1 and later. 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 the registered procedure rp_version is executed
- *ctwait.c* – implemented with Client-Library, registers with the Open Server to be notified when the registered procedure rp_version is executed

secsrv.c **sample program**

The *secsrv.c* sample program 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.

For more information on security services, refer to the Open Client and Open Server *Configuration Guide* for UNIX.

sigalarm.c **sample program**

The *sigalarm.c* sample program 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.

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.

This chapter covers the following topics:

Topic	Page
General instructions	59
Building an Embedded SQL/C executable	60
Using Embedded SQL/C sample programs	69

General instructions

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

- Set the following 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. Refer to the Sybase Adaptive Server® Enterprise *Installation Guide* for information on 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.

The following sections describe these steps.

Precompiling the application

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

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*]
  [@*options_file*]...
  [-a] [-b] [-c] [-d] [-e] [-f] [-h] [-l] [-m] [-p] [-r] [-s] [-u] [-v] [-w] [-x] [-y]
  filename[.ext]

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

64-bit applications
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. These precompilers can be used 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. These features are "off" by default, and are turned "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

This section gives information on libraries, linking, and header files.

Note Client-Library and Server-Library now support dynamic loading of Net-Library™, directory, and security drivers. This change affects the way you link Client-Library, Server-Library, and Embedded SQL applications.

You no longer need to explicitly link the following object files with your applications:

- Sybase Net-Library drivers
- Sybase directory drivers
- Sybase 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 the UNIX operating system.

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
Sun Solaris 2.x	`/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`
Sun Solaris 10 x64 (Opteron) 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 RS/6000	`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 9000 (8xx)	`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 Itanium	`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	`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 on POWER	`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 AMD64 (Opteron)/ EM64T	`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

Sun Solaris 2.x	`/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`

Sun Solaris 10 x64 (Opteron) 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 RS/6000	`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 9000 (8xx)	`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 Itanium	`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	`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 on POWER	`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 AMD64 (Opteron)/ EM64T	`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
Sun Solaris 2.x	`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`
Sun Solaris 10 x64 (Opteron) 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 RS/6000	`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 9000 (8xx)	`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 Itanium	`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	`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 on POWER	`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 AMD64 (Opteron)/EM64T	`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* in 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 non-threaded 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. For more information about building and linking a 64-bit application, see the sybopts.sh script available in the *$SYBASE/$SYBASE_OCS/sample/esqlc* directory.

For 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-UX will 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 will be able to 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 for more information.

Additional considerations

You should 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 eight-byte boundary, that is, to memory addresses that are multiples of eight bytes. Similarly, the data structure of 32-bit applications must be aligned on a four-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 a description of isql.

Using Embedded SQL/C sample programs

The Embedded SQL/C precompiler has two 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 prior to attempting to use the sample programs.

Read 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 error-handling and special-case-handling.

Location

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

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, you must 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, you must edit *sybsqlex.h* and change user name to your Adaptive Server login name and “password” to your Adaptive Server password.

Example 1: Using cursors for database query

The *example1.cp* sample program 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

Example 2: Displaying and editing rows of a table

The *example2.cp* sample program 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: Using cursors for database query with HA-Failover

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

Uni_example1: Using cursors for database query with unichar/univarchar support

The *uni_example1.cp* sample program shows how you can 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: Displaying and editing rows of a table with unichar/univarchar support

The *uni_example2.cp* sample program 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.

This chapter covers the following topics:

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. Refer to the Adaptive Server Enterprise *Installation Guide* for information on installing the pubs2 database.
- Set the following 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

This section gives information on libraries, linking, and the header files.

Libraries

Table 5-1 lists libraries that you should include if you want 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)
Sun 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-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 9000 (8xx) and HP Itanium	*libcl* – HP Transport Control Layer (system) *libBSD* – The BSD library (system) *libc_r* – C reentrant library *libdld* – (system)
IBM 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.

These steps are described in the following sections.

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]
```

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

64-bit applications
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.

Some of the options are switches that activate features of the precompiler, such as generating stored procedures. These features are "off" by default, and are turned "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 non-debug libraries.

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

Platform	Command
Sun Solaris 2.x	`cob -x program.cbl -L $SYBASE/$SYBASE_OCS/lib -lsybcobct -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lnsl -ldl -lm -lsocket -o program`
Sun Solaris 10 x64 (Opteron) 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 9000 (8xx)	`cob -x program.cbl -L $SYBASE/$SYBASE_OCS/lib -lsybcobct -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lBSD -lcl -lm -o program`
HP Itanium	`cob -x program.cbl -L $SYBASE/$SYBASE_OCS/lib -lsybct -lsybcobct -lsybtcl -lsybcs -lsybcomn -lsybintl -lsybunic -lcl -lm -ldld -o program`
IBM RS/6000	`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
Sun Solaris 2.x	`cob -g -x program.cbl -L $SYBASE/$SYBASE_OCS/devlib -lsybcobct -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lnsl -ldl -lm -lsocket -o program`
Sun Solaris 10 x64 (Opteron) 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 9000 (8xx)	`cob -g -x program.cbl -L $SYBASE/$SYBASE_OCS/devlib -lsybcobct -lsybct -lsybcs -lsybtcl -lsybcomn -lsybintl -lsybunic -lcl -lm -o program`
HP Itanium	`cob -g -x program.cbl -L $SYBASE/$SYBASE_OCS/devlib -lsybct -lsybcobct -lsybtcl -lsybcs -lsybcomn -lsybintl -lsybunic -lcl -lm -ldld -o program`
IBM RS/6000	`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 can result in a build error or can 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 available in the *SYBASE/$SYBASE_OCS/sample/esqlcob* directory.

Additional considerations

You should consider the following alignment issues when compiling and linking your application.

Data alignment on a 64-bit architecture

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

The following example illustrates this concept by creating a 32-bit and 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

Sybase version of the SQLDA

This code snippet shows the SQLDA layout that is 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;
```

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

- Elements defined as S9(4) – S9(4) is the ESQL/COBOL equivalent of smallint, which is two 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 four bytes.
- Elements defined as S9(9) – S9(9) is the ESQL/COBOL equivalent of an int, which is four bytes in length. Elements defined as S9(9) meet the 32-bit data alignment requirement.
- FILLER – a filler record two bytes in length is added to pad SD-SQLIND, which is an unpaired S9(4) element, and to align the entire structure on a four-byte boundary.

64-bit ESQL/COBOL version of the Sybase-specific SQLDA

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 eight-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 given above, the PIC clauses of relevance are:

- Elements defined as S9(4) – S9(4) is equivalent to an ESQL/COBOL smallint, which is two 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 eight. 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 four bytes, thus, a filler four bytes in length is added after SD-SQLD.
- Elements defined as S9(9) – S9(9) is equivalent to an ESQL/COBOL int, which is four bytes in length. A pair of S9(9) 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 eight 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 eight-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 prior to attempting to use 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. Read the *README* file for complete instructions on running the sample programs.

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 the *$SYBASE/$SYBASE_OCS/sample/esqlcob* directory:

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.

Example 1: Using cursors for database query

The *example1.pco* sample program shows how to use cursors in an interactive query program. Following is the sample’s program flow. 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 the program when Return is pressed at a prompt

Example 2: Displaying and editing rows in a table

The *example2.pco* sample program 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 on bcp, cpre, cobpre, defncopy, isql, installjava, and extractjava 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.	89
cpre	C precompiler utility, which precompiles a C source program to produce target, listing, and isql files.	114
cobpre	COBOL precompiler utility, which precompiles a COBOL source program to produce target, listing, and isql files.	125
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.	135
isql	Interactive SQL parser, which connects to and queries an Adaptive Server or Open Server.	140
installjava	Install java utility, which installs a JAR from a client file into an Adaptive Server.	158
extractjava	Extract java utility, which copies a retained JAR and the classes it contains from an Adaptive Server into a client file.	162

bcp

Description

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

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*]
[-M*LabelName 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.

datafile

The full path name of an operating system file. The path name can be from 1 to 255 characters in length. For multiple datafiles, use a comma-separated list of file names. For multiple datafiles and partitions, the number of datafiles 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.

The following error message appears if the character translation file(s) named with the -a parameter is missing, or you mistype the name(s):

```
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 the batch size to 1 causes Adaptive Server to allocate one data page to one row copied in. This parameter only applies to fast bcp, and is only useful in locating corrupt rows of data. Use -b 1 with care – doing so causes a new page to be allocated for each row, and is a poor use of space.

-c

Performs the copy operation with 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 the -d *discardfileprefix* option in conjunction with the -e *errorfile* to help identify and diagnose the problem rows logged 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 the bcp program 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 filename information for the error is added to the error file.

Sybase recommends that you use the -e *errorfile* option in conjunction with the -d *discardfileprefix* to help identify and diagnose the problem rows logged 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 do not have to duplicate your previous format responses interactively. Use this parameter only if you previously created a format file that you want to use now for a copy in or out. If this option is not used, bcp queries you for format information interactively.

-F *firstrow*

The number of the first row to copy (default is the first row). If multiple files are used, this option applies to each file.

Avoid using this parameter when performing heavy-duty, multi-process 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 --F cannot co-exist 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. The Standard Input is used as 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*, located 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. It specifies a DCE keytab file that contains the security key for the user name specified with -U option. Keytab files can be created with the DCE dcecp utility. See your DCE documentation for more information.

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 (default is the last row). If multiple files are used, 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.

When multiple partitions are used, this number will be 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 will be 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 your 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 your 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 this option 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 will 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 are unable to rerun 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 place holder 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. The Standard Output is used as 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 version 10.0.4 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 that your DSQUERY environment value specifies.

-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 Enterprise, the SDK, or Open Server 64-bit products with other Sybase 32-bit products overwrites the 32-bit binaries. Starting with ASE 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 to retain the peaceful coexistence of multiple products in the same Sybase installation. Since 32-bit binaries are released 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, you can 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 the following:

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

This returns a string like the following:

```
Sybase Client-Library/15.0/P-EBF14602 ESD
#9/DRV.15.0.3/SPARC/Solaris 8/BUILD1500-
099/64bit/OPT/Thu May 24 19:18:39 2007
```

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

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

This returns a string like the following:

```
Sybase Server-Library/15.0/P-EBF14603-14602 ESD
#9/DRV.15.0.3/SPARC/Solaris 8/BUILD1500-
099/64bit/OPT/Thu May 24 19:19:49 2007
```

-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 of key-letter options to enable additional security services. These key letters are:

- 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 will be set to CS_FALSE, and plain text (unencrypted) passwords will not be used in retrying the connection.

Note The -W option and the CS_SEC_NON_ENCRYPTION_RETRY property are ignored in this release.

-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 crashes, 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 IN.

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 -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 the *$SYBASE/$SYBASE_OCS/config* directory. 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 on 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 ASE. This does not automatically apply passwords to other encrypted columns, even if the second column is encrypted with the same key. The password must be supplied a second time to access the second column.

--hide-vcc

Instructs bcp not to copy virtual computed columns (VCC) either to or from a datafile. 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 this option is used, Adaptive Server does not calculate or send virtual computed column data.

--initstring *"TSQL_command"*

Sends Transact-SQL commands to ASE before data is transferred.

Result sets issued by the initialization string are silently ignored, unless an error occurs. If ASE 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 ASE.

--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 this parameter is not specified, 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, an error message displays.

Note --skiprows cannot co-exist 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 is printed:

```
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]:
```

In UNIX, 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 after bcp is finished.

Example 12 Sets the password to pwd1 for 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:

```
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 external OS file 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. It is capable of reading or writing 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.
- The bcp utility 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 violate 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.
- Functional indexes can be copied in and out of the Adaptive Server database using the --show-fi parameter.
- The data from the Virtual computed columns (VCC) can be eliminated from copying in and out of the Adaptive Server database using the --hide-vcc option.

Note To restrict the functionality of bcp to that of a previous version of bcp, you must 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.

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, the bcp logs all the rows from the beginning of the batch until the failed row.

Note If the discard file option is specified, the batch size is automatically adjusted and the message `Warning!!! Batch size adjusted to the value` ***`newbatchsize`***`, for the optimization of the discard file feature.` is displayed, when:

- -b *batchsize* is specified but the batch or row size is too big to hold all the rows of the batch in memory.
- The -b *batchsize* option is not specified.

Copying tables with indexes or triggers

- The bcp program 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 the insertion of rows is 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 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 the system procedure sp_dboption, "DB", 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 do not need to 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 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 are not allowed to 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 will not be able to recover your data.

- Unlogged bcp runs slowly while a dump database is taking place.
- Table A-1 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-1: Comparing fast and slow bcp

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

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

Note The performance penalty for copying data into a table that has indexes or triggers in place can be severe. If you are copying in a very large number of rows, it may be faster to drop all the indexes and triggers beforehand with drop index (or alter table for indexes created as a unique constraint) and drop trigger; set the database option; copy the data into the table; recreate the indexes and triggers; and then dump the database. However, you need to allocate 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 following data
  - 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-2 shows the defaults and possible alternate responses:

Table A-2: 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 the following: \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. For more information on datatype conversions, 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 releases have 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 can not identify which tab represents the end of the string. Since 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 less 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-3 and Table A-4 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-3: 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-4: Other datatypes converted to char storage

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

- Note that 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. (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/or prefixes.
- When copying data into a table, bcp observes any defaults defined for columns and user-defined datatypes. However, bcp ignores rules in order 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 row of data.
- 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 that is in character format from a file into a database table, check the datatype entry rules in the "Datatypes" section of the Adaptive Server Enterprise *Reference Manual*. Character data that is being copied into the database with bcp must conform to those rules. Note especially that 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(s) named with the -q parameter is missing, or you mistyped the name(s).

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.

64-bit applications
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. These precompilers can be used 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*]

```
 [-Uuser_id]
 [-Vversion_number]
 [-Zlanguage_locale_name]
[-a] [-b] [-c] [-d] [-e] [-f] [-h] [-l] [-m] [-p] [-r] [-s] [-u] [-v] [-w] [-x] [-y]
filename[.ext]
```

Note Options can be flagged using either a slash (/) or a dash (-);therefore, 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 when you want to check SQL semantics at precompile time. If -G is specified, a use *database* command will be added to the beginning of the *filename.sql* file. If you do not use this option, the precompiler uses your 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*] (argument is optional)
Generates stored procedures for appropriate SQL statements and saves them to a file for input to the database through isql. 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 name(s) will be the input file name(s) with the extension .isql appended (or replacing any input file name extension).

Also, see option -Ttag_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 HA-Failover capability.

-I *include_path_name*
Specifies a directory complete with the path name, where Embedded SQL will search 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 the following order:

- LC_ALL
- 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. The choices are:

- NONE
- SYNTAX
- SEMANTIC

NONE is the default value. If you use either SYNTAX or SEMANTIC, you must also specify the -U, -P, -S, and -D options so 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* (argument is optional)
Generates one or more listing files. A listing file is a version of the input file with each line numbered and followed by any applicable error message. 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 name(s) will be the input file name(s) with the extension *.lis* appended (or replacing any 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 may not use this option (default target file names will be assigned). If you do not use this option, the default target file name will be the input file name with the extension *.cbl* appended (or replacing any 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, then 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 type `-Tdbg` as part of your command, your generated stored procedures will be given 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 option -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_150 for Open Client and Open Server version 15.0).

-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 in the following order:

   - LC_ALL
   - 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."

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 as follows:

- For the 11.1 and later versions of cpre, the norebind attribute applies to all fetch statements of a cursor whose declaration was precompiled with the /b option.
- For the 10.0 and later versions of cpre, 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. For this option to work properly, 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. Refer to 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 sticky binds is enabled 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 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, the 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 only performed 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 you want as long as it does not violate any rules.

Examples

1 Run the precompiler (ANSI-compliant):

```
cpre program.pc
```

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

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

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

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

4 Display the precompiler version information only:

```
cpre -v
```

5 Run 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.
- Option format:
  Options will work with or without a space before the argument.
  For example, either of these formats will work:

  -Tdbg
  or
  -T dbg

- The precompiler can handle multiple input files. However, you may not use the option -O *target_file_name*, but must accept the default target file names (see "Target file" above). If you use option -G[isql_file_name], you cannot specify an argument; the default isql file names will be *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 will be *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 run-time error and aborts the application in use. You can avoid these issues by using -u with *cpre*. You may 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. These steps must be performed 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 with this command:

```
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 with this command:

```
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.

Following is the syntax for choosing a server on the precompile command line:

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

As an alternative, you can leave the server name out of 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 on DSQUERY.

cpre defaults

Table A-5 lists the options and defaults for the cpre and cobpre utilities:

Table A-5: 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 and later CS_VERSION_150 for version 15.0 and later
-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 available in the *$SYBASE/$SYBASE_OCS/bin* directory.

64-bit applications
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]]
    [-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 Options can be flagged using either a slash (/) or a dash (-);therefore, cobpre -I and cobpre /I 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 when you want to check SQL semantics at precompile time. If /G is specified, a use *database* command will be added to the beginning of the *filename.sql* file. If you do not use this option, the precompiler uses your 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*] (argument is optional)

Generates stored procedures for appropriate SQL statements and saves them to a file for input to the database through isql. 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 name(s) will be the input file name(s) with the extension *.isql* appended (or replacing any input file name extension).

Also, see option -Ttag_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 will search 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 the following order:

- LC_ALL
- 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. The choices are:

- NONE
- SYNTAX
- SEMANTIC

NONE is the default value. If you use either SYNTAX or SEMANTIC, you must also specify the -U, -P, -S, and -D options so 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* (argument is optional)
Generates one or more listing files. A listing file is a version of the input file with each line numbered and followed by any applicable error message. 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 name(s) will be the input file name(s) with the extension *.lis* appended (or replacing any 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 may not use this option (default target file names will be assigned). If you do not use this option, the default target file name will be the input file name with the extension *.cbl* appended (or replacing any 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, then 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 type `-Tdbg` as part of your command, your generated stored procedures will be given 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 option -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_150 for Open Client and Open Server version 15.0).

-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 in the following order:

   - LC_ALL
   - 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.”

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 as follows:

- For the 11.1 and later versions of cobpre, the norebind attribute applies to all fetch statements of a cursor whose declaration was precompiled with the -b option.
- For the 10.0 and later 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. For this option to work properly, 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. Refer to 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 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, the 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 only performed 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 you want as long as it does not violate any rules.

Examples

1. Run the precompiler (ANSI-compliant):

```
cobpre program.pco
```

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

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

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

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

4 Display the precompiler version information only:

```
cobpre -v
```

5 Run 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.
- Option format:
  Options will work with or without a space before the argument.
  For example, either of these formats will work:

  -Tdbg
  or
  -T dbg
- The precompiler can handle multiple input files. However, you may not use the option -O *target_file_name*, but must accept the default target file names (see “Target file” above). If you use option -G[isql_file_name], you cannot specify an argument; the default isql file names will be *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 will be *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 run-time error and aborts the application in use. You can avoid these issues by using -u with *cobpre*. You may 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. These steps must be performed 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 with this command:

```
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 with this command:

```
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.

Following is the syntax for choosing a server on the precompile command line:

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

As an alternative, you can leave the server name out of 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 on DSQUERY.

cobpre | cpre defaults

See Table A-5 for a list of options and defaults for the 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 available in the *$SYBASE/$SYBASE_OCS/bin* directory.

Note The defncopy utility 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 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.

Note The ascii_7 character set is compatible with all character sets. If either the Adaptive Server's or client's character set is set to ascii_7, any 7-bit ASCII character is allowed to pass between client and server unaltered. Other characters produce conversion errors. Character set conversion issues are covered more 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. (See the Adaptive Server Enterprise *System Administration Guide* for more information about character sets and the associated flags.

-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 and returns to the operating system.

SDK binaries like defncopy have the same names in both the 32-bit and 64-bit products. Installing Adaptive Server Enterprise, the SDK, or Open Server 64-bit products with other Sybase 32-bit products overwrites the 32-bit binaries. Starting with ASE 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 to retain the peaceful coexistence of multiple products in the same Sybase installation. Since 32-bit binaries are released 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, you can 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 the following:

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

This returns a string like the following:

```
Sybase Client-Library/15.0/P-EBF14602 ESD
#9/DRV.15.0.3/SPARC/Solaris 8/BUILD1500-
099/64bit/OPT/Thu May 24 19:18:39 2007
```

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

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

This returns a string like the following:

```
Sybase Server-Library/15.0/P-EBF14603-14602 ESD
#9/DRV.15.0.3/SPARC/Solaris 8/BUILD1500-
099/64bit/OPT/Thu May 24 19:19:49 2007
```

-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 parameter; any password supplied with the -P parameter is ignored.

-V can be followed by a *security_options* string of key-letter options to enable additional security services. These key letters are:

- 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 the defncopy crashes, 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 at installation, or afterwards with 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. For more information on security mechanism names, 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 name(s) of database object(s) for defncopy to copy out. Do not use *object_name* when copying definitions in.

owner
: Specifying this 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 are displayed 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 non-interactive way of copying 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 filenames that reflect both the object's name and its owner.
- defncopy ends each definition that it copies out with the following comment:

  ```
  /* ### 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 available in the *$SYBASE/$SYBASE_OCS/bin* directory.

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*]
[-V [*security_options*]]
[-w *column_width*]
[-x *trusted.txt_file*]
[-y *sybase_directory*]
[-z *localename*]
[-Z *security_mechanism*]
[--conceal [':?' | '*wildcard*']]
[--help]
[--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 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.

Note The ascii_7 character set is compatible with all character sets. If either the Adaptive Server's or client's character set is set to ascii_7, any 7-bit ASCII character is allowed to pass between client and server unaltered. Other characters produce conversion errors. Character set conversion issues are covered more 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 is displayed 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 Adaptive Server's 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. Make sure to escape shell meta-characters 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 non-standard 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. It specifies a DCE keytab file that contains the security key for the user name specified with -U option. Keytab files can be created with the DCE dcecp utility. See your DCE documentation for more information.

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 *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 display; 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 will be 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 your 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 your 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 in conjunction 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 from 6 to 30 characters in length. If your password is NULL, use -P without any password.

-Q

Provides clients with failover (HA) property. See the Adaptive Server Enterprise *Using Sybase Failover in a High Availability System* for more information.

-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 is blank by default. 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 into isql is 60 seconds.

-U *username*
Specifies a login name. Login name is case sensitive.

-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 of key-letter options to enable additional security services. These key letters are:

- 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 Enterprise, the SDK, or Open Server 64-bit products with other Sybase 32-bit products overwrites the 32-bit binaries. Starting with ASE 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 to retain the peaceful coexistence of multiple products in the same Sybase installation. Since 32-bit binaries are released 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, you can 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 the following:

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

This returns a string like the following:

```
Sybase Client-Library/15.0/P-EBF14602 ESD
#9/DRV.15.0.3/SPARC/Solaris 8/BUILD1500-
```

```
099/64bit/OPT/Thu May 24 19:18:39 2007
```

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

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

This returns a string like the following:

```
Sybase Server-Library/15.0/P-EBF14603-14602 ESD
#9/DRV.15.0.3/SPARC/Solaris 8/BUILD1500-
099/64bit/OPT/Thu May 24 19:19:49 2007
```

-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 will be set to CS_FALSE, and plain text (unencrypted) passwords will not be used in retrying the connection.

Note The -W option and the CS_SEC_NON_ENCRYPTION_RETRY property are ignored in this release.

-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 is used as the first preference.

If isql crashes, 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 at installation, or afterward with 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. For more information on security mechanism names, see the description of the *libtcl.cfg* file in the Open Client and Open Server *Configuration Guide* for UNIX.

--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.

For information on how to use the wildcard in an isql session, see "Using prompt labels and double wildcards in an isql session" on page 157.

--help

Displays a brief description of syntax and usage for the isql utility consisting of a list of available arguments.

--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 ASE error number to stderr, and returns a value of "2" to the calling program. As before, isql prints the full server error message to stdout.

Examples

Example 1 Puts you in a text file 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, 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>
```

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 ouput 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 build using the non-threaded 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, give the command 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 display 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
  ```

- Case is significant for the isql flags.

- 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 with the following command:

  ```
  :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 with the following command:

  ```
  :R filename
  ```

- When using isql interactively, you can change the current database with the following command:

  ```
  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:

  ```
  /*
  **go
  */
  ```

  Do not use the following:

  ```
  /*
  go
  */
  ```

- isql defines the order of the date format as month, date, and year (mm dd yyy hh:mm AM or PM), regardless of the locale environment. To change this default order, user the convert function.

Additional commands within isql:

Table A-6: 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 these characters are placed at the beginning of an isql line.

See also

sp_addlanguage, sp_addlogin, sp_configure, sp_defaultlanguage, sp_droplanguage, and sp_helplanguage in the Adaptive Server Enterprise *Reference Manual*.

installjava

Description

Installs a JAR from a client file into an Adaptive Server. This utility is available in the *$SYBASE/$SYBASE_OCS/bin* directory.

Syntax

```
installjava
    -f file_name
    [ -new | -update ]
    [ -j jar_name ]
    [ -S server_name ]
    [ -U user_name ]
    [ -P password ]
    [ -D database_name ]
    [ -I interfaces_file ]
    [ -a display_charset ]
    [ -J client_charset ]
    [ -z language ]
    [ -t timeout ]
    [-v]
```

Parameters

-f *file_name*

The name of the source file containing the classes to be installed in the database.

-new | -update

Specifies whether the classes in the file already exist in the database.

If you specify the new parameter, you cannot install a class with the same name as an existing class.

If you specify the update parameter, you can install a class with the same name as an existing class, and the newly installed class replaces the existing class.

-j *jar_name*

The name of the JAR containing the classes to be installed in the database. Indicates that the JAR file should be saved in the database and associated with the classes it contains.

-S *server_name*

The name of the server.

-U *user_name*

An Adaptive Server login name. If you omit the -U flag and parameter, or if you specify the -U flag with no parameter, Adaptive Server uses the current user's operating system login name.

-P *password*
An Adaptive Server password. If you omit the -P flag and parameter, installjava prompts for a password. If you specify the -P flag with no password, the null password is used.

-D *database_name*
The name of the database in which to install the JAR. If you omit the -D flag, or if you specify the -D flag with no parameter, the user's default database is used.

-I *interfaces_file*
The name and location of the interfaces file to search when connecting to Adaptive Server. If you omit the -I flag and parameter, or if you specify the -I flag with no parameter, the interfaces file in the directory designated by your SYBASE environment variable is used.

-a *display_charset*
Allows you to use installjava from a machine where the character set differs that of the server. Use -a 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.

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

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

-J with no argument disables character set conversion. 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. See the Adaptive Server Enterprise *System Administration Guide* for more information about character sets and associated flags.

-z *language*
The name of an alternate language for displaying installjava prompts and messages. Without the -z flag, installjava uses the server's default language. You can add languages to an Adaptive Server during installation or afterward, using the langinstall utility or the sp_addlanguage stored procedure.

-t *timeout*

Specifies the number of seconds before a SQL command times out. If you do not specify a timeout, the command runs indefinitely. This affects commands issued from within installjava, not the connection time. The default timeout for logging into installjava is 60 seconds.

-v

Prints the version number and copyright message for installjava and then exits.

Examples

Example 1 Installs *addr.jar* and its classes, but does not retain the association between the JAR and classes:

```
installjava -f '/home/usera/jars/addr.jar' -new
```

Example 2 Reinstalls *addr.jar* and associates its classes with the employees JAR name:

```
installjava -f '/home/usera/jars/addr.jar' -update -j employees
```

Usage

- You must set the SYBASE environment variable to the location of the current version of Adaptive Server before you can use installjava.
- Refer to *Java in Adaptive Server Enterprise* for more information about how this utility is used when Java is enabled in the database.
- Any user can reference installed classes.
- The parameter flags -f, -j, -S, -U, -P, -D, and -I can be written with or without a space between the flag letter and the following parameter.

Adding new JARs

- If you use new with the -jar option and a JAR of that name already exists in the database, an exception is raised.
- If any classes of the same name as those in the source JAR already exist in the database, an exception is raised.

Updating JARs and classes

Warning! If you alter a class used as a column datatype by reinstalling a modified version of the class, you must make sure that the modified class can read and use existing objects (rows) in tables using that class as a datatype. Otherwise, you may be unable to access those objects without reinstalling the class.

- If you use -update with the -jar option:

    - All classes in the database associated with the target JAR are deleted from the database and the classes in the source JAR file installed in their place.
    - If a class in the source JAR file is already installed in the database but is not attached to a JAR, the class in the source JAR is installed in the database and the unattached class is deleted.
- If you use -update without the -jar option:
    - Classes in the source JAR file replace unattached classes of the same name.
    - Classes in the source JAR that do not correspond to an installed class are installed as unattached classes in the database.
- If you install a new JAR with a replacement for an installed class that is referenced by a SQLJ procedure or function, make sure that the newly installed class has a valid signature for the SQLJ routine. If the signature is invalid, an exception is raised when the SQLJ routine is invoked.

Locks

- When you execute installjava, an exclusive lock is placed on sysxtypes.
- If -jar is specified, an exclusive table lock is placed on sysjars.

Permissions

You need to be a System Administrator or Database Owner to use installjava.

Tables used

sysjars, sysxtypes

See also

Commands remove java

System procedures sp_helpjava

Utilities extractjava

extractjava

Description

Copies a retained JAR and the classes it contains from an Adaptive Server into a client file. This utility is available in the *$SYBASE/$SYBASE_OCS/bin* directory.

Syntax

```
extractjava
    -j jar_name
    -f file_name
    [-S server_name]
    [-U user_name]
    [-P password]
    [-D database_name]
    [-I interfaces_file]
    [-a display_charset]
    [-J client_charset]
    [-z language]
    [-t timeout]
    [-v]
```

Parameters

-j *jar_name*
: The name assigned to the retained JAR in the database that is the source of the transfer.

-f *file_name*
: The name of the client file that is the target of the transfer.

-S *server_name*
: The name of the server.

-U *user_name*
: An Adaptive Server login name. If you omit the -U flag and parameter, or if you specify the -U flag with no parameter, Adaptive Server uses the current user’s operating system login name.

-P *password*
: An Adaptive Server password. If you omit the -P flag and parameter, extractjava prompts for a password. If you specify the -P flag with no password, the null password is used.

-D *database_name*
: The name of the database in which to install the JAR. If you omit the -D flag, or if you specify the -D flag with no parameter, the user’s default database is used.

-I *interfaces_file*
The name and location of the interfaces file to search when connecting to Adaptive Server. If you omit the -I flag and parameter, or if you specify the -I flag with no parameter, the interfaces file in the directory designated by your SYBASE environment variable is used.

-a *display_charset*
Allows you to use extractjava from a machine where the character set differs that of the server. Use -a 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.

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

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

-J with no argument disables character set conversion. 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. See the Adaptive Server Enterprise *System Administration Guide* for more information about character sets and associated flags.

-z *language*
The name of an alternate language for displaying extractjava prompts and messages. Without the -z flag, extractjava uses the server's default language. You can add languages to an Adaptive Server during installation or afterward, using the langinstall utility or the sp_addlanguage stored procedure.

-t *timeout*
Specifies the number of seconds before a SQL command times out. If you do not specify a timeout, the command runs indefinitely. This affects commands issued from within extractjava, not the connection time. The default timeout for logging into extractjava is 60 seconds.

-v
prints the version number and copyright message for extractjava and then exits.

Examples

Downloads the classes associated with the employees JAR to the client file *newaddr.jar*.

- *UNIX*:

```
extractjava -j employees -f '/home/usera/jars/addr.jar' -new
```

Usage

- You must set the SYBASE environment variable to the location of the current version of Adaptive Server before you can use extractjava.
- If the target client file already exists, extractjava overwrites its contents.
- The parameter flags -f, -j, -S, -U, -P, -D, and -I can be written with or without a space between the flag letter and the following parameter.
- When you execute extractjava, an exclusive lock is placed on sysxtypes.
- If -jar is specified, an exclusive table lock is placed on sysjars.
- See the Adaptive Server Enterprise *Java in Adaptive Server Enterprise* for more information about how this utility is used when Java is enabled in the database.

Permissions

You need to be a System Administrator or Database Owner to use extractjava.

Tables used

sysjars, sysxtypes

See also

Commands remove java

System procedures sp_helpjava

Utilities installjava

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 depends 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*.
- 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. Refer to 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 9000 (8xx) - 32-bit	hpux	SHLIB_PATH
HP 9000 (8xx) - 32-bit using native threads	nthread_hpux	
HP 9000 (8xx) - 64-bit	hpux64	LD_LIBRARY_PATH
HP 9000 (8xx) - 64-bit using native threads	nthread_hpux64	
HP Itanium - 32-bit	hpia	SHLIB_PATH
HP Itanium - 32-bit using native threads	nthread_hpia	
HP Itanium - 64-bit	hpia64	LD_LIBRARY_PATH
HP Itanium - 64-bit using native threads	nthread_hpia64	
IBM RS/6000 - 32-bit	rs6000	LIBPATH
IBM RS/6000 - 32-bit using native threads	nthread_rs6000	
IBM RS/6000 - 64-bit	rs600064	LIBPATH
IBM RS/6000 - 64-bit using native threads	nthread_rs600064	
Linux - 32-bit	linux	LD_LIBRARY_PATH
Linux - 32-bit using native threads	nthread_linux	
Linux on POWER - 32-bit	ibmplinux	LD_LIBRARY_PATH
Linux on POWER - 32-bit using native threads	nthread_ibmplinux	
Linux on POWER - 64-bit	ibmplinux64	LD_LIBRARY_PATH
Linux on POWER - 64-bit using native threads	nthread_ibmplinux64	
Linux AMD64 (Opteron)/EM64T	linuxamd64	LD_LIBRARY_PATH
Linux AMD64 (Opteron)/EM64T using native threads	nthread_linuxamd64	
Sun Solaris 2.x - 32-bit	sun_svr4	LD_LIBRARY_PATH
Sun Solaris 2.x - 32-bit using native threads	nthread_sun_svr4	

Platform	SYBPLATFORM setting	Platform-specific library path variable
Sun Solaris 2.x - 64-bit	sun_svr464	LD_LIBRARY_PATH_64
Sun Solaris 2.x - 64-bit using native threads	nthread_sun_svr464	
Sun Solaris 10 x64 (Opteron) 32-bit	sunx86	LD_LIBRARY_PATH
Sun Solaris 10 x64 (Opteron) 32-bit using native threads	nthread_sunx86	
Sun Solaris 10 x64 (Opteron) 64-bit	sunx8664	LD_LIBRARY_PATH_64
Sun Solaris 10 x64 (Opteron) 64-bit using native threads	nthread_sunx8664	

For Embedded SQL/COBOL applications you must set the following environment variables in addition to the ones listed above:

- COBDIR – set to the path of your COBOL compiler
- PATH – add $COBDIR/bin

Index