New Features Open Server™ 15.0 and SDK 15.0 for Microsoft Windows, Linux, UNIX, and Mac OS X

Document ID: DC20155-01-1500-23

Last revised: July 2009

This document describes new features available for Open Server™ 15.0 and the Software Developer's Kit (SDK) 15.0. It is revised to include new features as they become available.

Copyright 2009 by Sybase, Inc. All rights reserved. Sybase trademarks can be viewed at the Sybase trademarks page at http://www.sybase.com/detail?id=1011207. Sybase and the marks listed are trademarks of Sybase, Inc. ® indicates registration in the United States of America. Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. Unicode and the Unicode Logo are registered trademarks of Unicode, Inc. All other company and product names mentioned may be trademarks of the respective companies with which they are associated.

Product platforms and compatibilities

Table 1 lists the platforms and the year Open Server and SDK were first built and released on these platforms:

Table 1: Platforms that support Open Server and SDK

Platform	Release date
SDK 15.0 on APPLE / Mac Intel	November 2008
Linux Red Hat 4.0 on IA64	September 2008
Windows 2003 (x64) Service Pack 1	April 2006
HP Itanium (HP-UX 11.23 or later) 32-bit and 64-bit	April 2006
Linux AMD64 (Opteron)/EM64T	April 2006
Sun Solaris 10 x64 32-bit and 64-bit	April 2006
Linux on POWER 32-bit and 64-bit	November 2005
HP-UX 11.11 32-bit and 64-bit	August 2005
IBM RS/6000 AIX 5.2 32-bit and 64-bit	August 2005
Linux x86 32-bit	August 2005
Sun Solaris 8 (SPARC) 32-bit and 64-bit	August 2005
Windows 2000 (x86) 32-bit Service Pack 4	August 2005

Note Not all Open Server and SDK components are available on the platforms listed above. See “Product components” on page 7 for the complete list of components available for your platform.

Table 2 lists the platforms, compilers and third-party products Open Server and SDK products are built and tested on:

Table 2: Open Client and Open Server platform compatibility matrix

Platform	Operating system level	C and C++ compilers	COBOL compilers	Kerberos version	Light-weight Directory Access (LDAP)	Secure Sockets Layer (SSL)
APPLE	SDK 15.0 32-bit on Mac OS X / Intel	gcc-4.0.1	Not available	Not available	Not available	Open SSL 0.9.7m
HP-UX 11.11 32-bit	HP-UX 11i	HP C 11.00.00 ANSI HP ANSI C++ B3910B A.06.00	MFServer Express 4.0 SP2	CyberSafe Trust Broker 2.1, MIT 1.4.1	OpenLDAP 2.3.27 including OpenSSL 0.9.8c	Certicom SSL Plus 5.2.2, SBGSE 2.2

Platform	Operating system level	C and C++ compilers	COBOL compilers	Kerberos version	Light-weight Directory Access (LDAP)	Secure Sockets Layer (SSL)
HP-UX 11.11 64-bit	HP-UX 11i with Patch bundle 99OP	HP C 11.00.00 HP ANSI C++ B3910B A.06.00	MF Server Express 4.0 SP2	MIT 1.4.3	OpenLDAP 2.3.27 including OpenSSL 0.9.8c	Certicom SSL Plus 5.2.2, SBGSE 2.2
HP Itanium 32-bit	HP-UX 11.23	C/C++ aCC HP aC++/ANSI C B3910B A.05.05	MF Server Express 4.0 SP2	MIT 1.4.1	OpenLDAP 2.3.27 including OpenSSL 0.9.8c	Certicom SSL5.2.2
HP Itanium 64-bit	HP-UX 11.23	C/C++ aCC HP aC++/ANSI C B3910B A.05.05	MF Server Express 4.0 SP2	MIT 1.4.1	OpenLDAP 2.3.27 including OpenSSL 0.9.8c	Certicom SSL5.2.2
IBM RS/6000 AIX 5.2 32-bit	AIX 5.2	C++ 5.0.22	MF Server Express 4.0 SP2	CyberSafe Trust Broker 2.1, MIT 1.4.1	OpenLDAP 2.3.27 including OpenSSL 0.9.8c	Certicom SSL Plus 5.2.2, SBGSE 2.2
IBM RS/6000 AIX 5.2 64-bit	AIX 5.2	C++ 5.0.22	MF Server Express 4.0 SP2	MIT 1.4.3	OpenLDAP 2.3.27 including OpenSSL 0.9.8c	Certicom SSL Plus 5.2.2, SBGSE 2.2
Linux x86 32-bit	Red Hat EL 3.0	gcc 3.2.3-42	MF Server Express 4.0 SP2	MIT 1.4.1	OpenLDAP 2.3.27 including OpenSSL 0.9.8c	Certicom SSL Plus 5.2.2
Linux on Itanium 64-bit	Red Hat EL 4.0 Update 4	gcc version 3.4.6-3 20060404 glibc-2.3.4-2.25	Not available	Not available	Not available	Not available
Linux on POWER 32-bit	Red Hat EL 3.0	xlc 7.0.0-1	Not available	MIT 1.4.1	OpenLDAP 2.3.27 including OpenSSL 0.9.8c	OpenSSL 0.9.71
Linux on POWER 64-bit	Red Hat EL 3.0	xlc 7.0.0-1	Not available	MIT 1.4.1	OpenLDAP 2.3.27 including OpenSSL 0.9.8c	OpenSSL 0.9.71
Linux AMD64 (Opteron)/ EM64T	Red Hat EL 3.0	gcc 3.x	Not available	MIT 1.4.3	OpenLDAP 2.3.27 including OpenSSL 0.9.8c	Certicom SSL Plus 5.2.2

Platform	Operating system level	C and C++ compilers	COBOL compilers	Kerberos version	Light-weight Directory Access (LDAP)	Secure Sockets Layer (SSL)
Sun Solaris 8 (SPARC) 32-bit	Solaris 8	Sun C/C++ 6.2	MF Server Express 4.0 SP2	CyberSafe Trust Broker 2.1, MIT 1.4.1	OpenLDAP 2.3.27 including OpenSSL 0.9.8c	Certicom SSL Plus 5.2.2, SBGSE 2.2
Sun Solaris 8 (SPARC) 64-bit	Solaris 8	Sun C/C++ 6.2	MF Server Express 4.0 SP2	CyberSafe Trust Broker 2.1, MIT 1.4.1	OpenLDAP 2.3.27 including OpenSSL 0.9.8c	Certicom SSL Plus 5.2.2, SBGSE 2.2
Sun Solaris 10 x64 32-bit	Solaris 10	Sun Studio 10	Not available	MIT 1.4.2	OpenLDAP 2.3.27 including OpenSSL 0.9.8c	Certicom SSL Plus 5.2.2
Sun Solaris 10 x64 64-bit	Solaris 10	Sun Studio 10	Not available	MIT 1.4.2	OpenLDAP 2.3.27 including OpenSSL 0.9.8c	Certicom SSL Plus 5.2.2
Windows 2000 (x86) Service Pack 4 (32-bit)	Windows 2000	MS C 6.0 (Microsoft Developers Studio; unoptimized, development only)	Net Express 4.0	CyberSafe Trust Broker 4.0, MIT 2.6.5	OpenLDAP 2.3.27 including OpenSSL 0.9.8c	Certicom SSL Plus 5.2.2, SBGSE 2.2
Windows 2003 64-bit	Windows 2003	C/C++ compiler 14.00.40310.41	Not available	Not available	OpenLDAP 2.3.27 including OpenSSL 0.9.8c	Open SSL 0.9.8d

Note For Open Server and SDK certifications support, see the Sybase platform certifications page at http://certification.sybase.com/ucr/search.do.

Product components

The following sections summarize the components of SDK and Open Server products and the platforms these components are supported on.

Open Server

Open Server is a set of APIs and supporting tools that you can use to create custom servers to respond to client requests submitted through Open Client™ or jConnect routines. Table 3 lists the Open Server components and the platforms these components are supported on.

Table 3: Open Server components and supported platforms

Open Server components	Platforms
Open Server Server-Library	All platforms
Open Server Client-Library	All platforms
Language modules	All platforms

Software Developer's Kit

Software Developer's Kit (SDK) is a set of libraries and utilities that you can use to develop client applications. Table 4 lists the SDK components and the platforms these components are supported on.

Table 4: SDK components and supported platforms

SDK components	Platforms
Open Client Client-Library	All platforms
Open Client DB-Library™	All platforms
Embedded SQL™/C (ESQL/C)	All platforms
Embedded SQL/COBOL (ESQL/COBOL)	• HP-UX Itanium 32-bit and 64-bit • HP-UX PA-RISC 32-bit and 64-bit • IBM AIX 32-bit and 64-bit • Linux x86 32-bit • Sun Solaris SPARC 32-bit and 64-bit • Windows x86 32-bit
Extended Architecture (XA)	• HP-UX PA-RISC 32-bit and 64-bit • IBM AIX 32-bit and 64-bit • Sun Solaris SPARC 32-bit and 64-bit • Windows x86 32-bit
jConnect™ for JDBC™	All platforms

SDK components	Platforms
Adaptive Server® Enterprise ODBC Driver by Sybase®	• Linux x86 32-bit • Linux x86-64 64-bit • Mac OS X Intel • Windows x86 32-bit • Windows x64 64-bit
Adaptive Server Enterprise OLE DB Provider by Sybase	• Windows x86 32-bit • Windows x64 64-bit
Adaptive Server Enterprise ADO.NET Data Provider	• Windows x86 32-bit • Windows x64 64-bit
Language modules	All platforms

SDK DB-Library Kerberos Authentication option

The Sybase SDK DB-Library Kerberos Authentication Option allows the MIT Kerberos security mechanism to be used on DB-Library and is available on:

- Linux x86 32-bit
- Sun Solaris SPARC 32-bit and 64-bit
- Windows x86 32-bit

IPv6 support

As of ESD #14, Open Server and SDK support IPv6 on all the platforms on which they are released, except SDK 15.0 on Mac OS X/Intel.

Character set support

The Adaptive Server Enterprise ODBC Driver, ADO.NET Data Provider, and OLE DB Provider communicate with Adaptive Server using a negotiated character set, which you can configure in the driver's user interface as User Specified, Server Default, or Client Default. Server Default is the default setting; however, ADO.NET, OLE DB, and ODBC do not support all character sets that Adaptive Server supports, and using an unsupported character set raises this error:

```
[Sybase][driver] Could not load code page for requested
charset
```

To avoid this error, perform one of the following:

- In your driver's user interface, go to the Advanced tab and select Client Default as your character set. If you choose Server Default or User Specified, ensure that the server's default character set or your specified character set is supported.
- In the connection string, ensure that the charset property specifies a supported character set.

Note Microsoft Windows x86 32-bit and Microsoft Windows x64 64-bit do not support roman8, roman9, and ascii_8.

Upgrading to the new version

To upgrade Open Server applications (srvlib):

- For statically linked applications, perform a complete rebuild of the applications with the new version of software. Recompile and relink the applications with the new header files and libraries.

- For dynamically linked applications, recompile and relink the SDK libraries that have changed to include “syb” in library names.

  Note If you have made any changes to the application files, you must recompile.

Ensure that the runtime libraries are for the same major release as the version used to build the application.

New feature ESD #19

This section describes DBPROP_MAXROWS support, which is the only new feature in ESD #19.

DBPROP_MAXROWS support

The Adaptive Server Enterprise OLE DB Provider now supports DBPROP_MAXROWS, a rowset property that you can use to limit the number of rows returned in a row set. The default value of this property is 0, which indicates no limit to the number of returned rows.

See the Microsoft Developer Network at http://msdn.microsoft.com.

New feature in ESD #18

This section describes isql command history, which is the only new feature in ESD #18.

isql command history

The isql command history feature enables you to list, recall, and reissue past commands. The command history is loaded into memory when you start isql, and is updated whenever a new command is issued. If specified, isql saves the in-memory history to disk when isql shuts down.

Note Command history is specific to an isql session and not to a connection; this allows you to track the command history even if the session is dynamically migrated to a different server.

Activating command history

By default, the command history feature is off. Activate it by using the --history command line option. --history loads the contents of the command history log file, if it exists, when isql starts.

Syntax

isql [--history [p]*history_length* [--history_file *history_filename*]]

Parameters

- p – indicates command history persistence; in-memory command history is saved to disk when isql shuts down. If you do not use the p option, the command history log is deleted after its contents are loaded into memory.
- *history_length* – this parameter, which is required if you use --history, is the number of commands that isql can store in the command history log. The maximum value of *history_length* is 1024; if a larger value is specified, isql silently truncates it to 1024.
- --history_file *history_filename* – indicates that isql must retrieve the command history log from *history_filename*. If p is specified, isql also uses *history_filename* to store the current session's command history. *history_filename* can include an absolute or a relative path to the log file. A relative path is based on the current directory. If you do not indicate a path, the history log is saved in the current directory.

  When --history_file is not specified, isql uses the default log file:

  - For UNIX: *$HOME/.sybase/isql/isqlCmdHistory.log*
  - For Windows: *%APPDATA%\Sybase\isql\isqlCmdHistory.log*

Examples

Example 1 Deletes *myaseHistory.log* after loading its contents to memory. The session's command history is not stored:

```
isql -Uguest -Ppassword -Smyase --history 1024
   --history_file myaseHistory.log
```

Example 2 Loads and saves the command history using the default log file:

```
isql -Uguest -Ppassword -Smyase --history p1024
```

Usage

- The command history feature is available only in command mode. Also, only commands that are issued interactively in isql are included in the command history. Examples of commands that are not included in the command history are those that are executed using the -i command line option or as part of a redirected input such as:

```
isql -Uguest -Ppassword -Smyase --history p1024
   --history_file myaseHistory.log <<EOF
exec sp_x_y_z
go
EOF
```

- Command history contains the most recent commands issued in an isql session. When *history_length* is reached, isql drops the oldest command from the history and adds the newest command issued.
- If you do not specify an alternate log file, and if the *$HOME* or *%APPDATA%* environment variable used by the default log file is not defined, an error message appears and the command history log is not saved.

Listing command history

In an isql session, use the h command to display the command history. A page can display up to 24 lines of commands. If the command history contains more than 24 lines, press Enter to display the next set of commands or enter “a” to display all commands in one page. Enter “q” to return to isql.

Syntax

h [*n*]

Parameter

n – indicates the number of commands to appear. If *n* is positive, the commands that appear start from the oldest command in the history. If *n* is negative, the *n* most recent commands appear.

Examples

Example 1 Lists all the commands stored in the command history:

```
1> h
[1] select @@version
[2] select db_name()
[3] select @@servername
1>
```

Example 2 Lists the two most recent commands issued:

```
1> h -2
```

```
[2] select db_name()
[3] select @@servername
1>
```

Recalling and reissuing commands

Use the ? command to recall and reissue a command from the command history.

Syntax

? *n* | ??

Parameter

- *n* – when *n* is positive, isql looks for the command labeled with the number *n* and loads this to the command buffer. When *n* is negative, isql loads the *n*th most recent command issued.
- ?? – recalls the latest command issued and is equivalent to ? -1.

Examples

In this sample command history used for examples 1 and 2, `[1]` tags the oldest command issued and `[3]` tags the most recent command issued:

```
[1] select @@version
[2] select db_name
[3] select @@servername
```

Example 1 Recalls the command labeled 1 from the command history:

```
1> ? 1
1> select @@version
2>
```

Example 2 Recalls the latest issued command from the command history:

```
1> ? -1
1> select @@servername
2>
```

Usage

- When a command is recalled from history, the recalled command overwrites the command in the command buffer.
- You can edit a recalled command before resubmitting the command to the server.

New features in ESD #17

This section describes the new features in ESD #17.

New *isql* command line option *--appname*

The new isql option --appname allows you to change the default application name *isql* to the isql client application name. This feature simplifies:

- Testing of Adaptive Server cluster routing rules for incoming client connections based on the client application name.
- Switching between alternative settings for isql in *$SYBASE/$SYBASE_OCS/config/ocs.cfg*, such as between debugging and normal sessions.
- Identification of the script that started a particular isql session from within Adaptive Server.

Syntax

isql --appname "*application_name*"

where *application_name* is the client application name. You can retrieve the client application name from sysprocesses.program_name after connecting to your host server.

application_name has a maximum length of 30 characters. You must enclose the entire application name in single quote or double quote characters if it contains any white spaces that do not use the backslash escape character. You can set the *application_name* to an empty string.

Note You can also set the client application name in *ocs.cfg* using the CS_APPNAME property.

Examples

Example 1 Sets the application name to "isql Session 01":

```
isql -UmyAccount -SmyServer --appname "isql Session 01"
Password:
1>select program_name from sysprocesses
2>where spid=@@spid
3>go


program_name
--------------------
isql Session 01
```

Example 2 Sets the application name to the name of the script that started the isql session:

```
isql --appname $0
```

Example 3 The following sample *ocs.cfg* file allows you to run isql normally or with network debug information. Because the configuration file is read and interpreted after the command line parameters are read and interpreted, setting CS_APPNAME to *isql* sets the application name back to isql:

```
;Sample ocs.cfg file
[DEFAULT]
;place holder

[isql]
;place holder

[isql_dbg_net]
CS_DEBUG = CS_DBG_NETWORK
;CS_APPNAME = "isql"
```

To run isql normally:

```
isql -Uguest
```

To run isql with network debug information:

```
isql -Uguest --appname isql_dbg_net
```

New CT-Library debug flag CS_DBG_SSL

The new CS_DBG_SSL flag allows your CT-Library application to write SSL-related diagnostics and error codes to standard output (stdout) or to a log file. CS_DBG_SSL does not require *devlib* libraries and can be used in isql. You can also use CS_DBG_SSL with normal libraries.

New environment variables for controlling CT-Library debugging

These environment variables are introduced in ESD #17 to help you debug your CT-Library applications without the need to modify and relink:

- SYBOCS_DEBUG_FLAGS – enables specific diagnostic subsystems. You can enable multiple debug options by specifying a comma-delimited list of flags in the variable.

- SYBOCS_DEBUG_LOGFILE – specifies the log file where the diagnostics are recorded. If you do not set SYBOCS_DEBUG_LOGFILE, messages are written to stdout.

Note Debug flags that require *devlib* libraries still require *devlib* libraries even when using SYBOCS_DEBUG_FLAGS. For information about which ct_debug flag parameters require *devlib* libraries, see "ct_debug" in the "Routines" chapter in the Open Client *Client-Library/C Reference Manual*.

To use, set the variables before calling your CT-Library application. For example:

On UNIX:

```
% setenv SYBOCS_DEBUG_FLAGS CS_DBG_SSL,CS_DBG_PROTOCOL
% setenv SYBOCS_DEBUG_LOGFILE libsybfssl.log
% ./isql –U sa –P –S my_ssl_server
% more libsybfssl.log
```

On Windows:

```
C:\> set SYBOCS_DEBUG_FLAGS=CS_DBG_SSL
C:\> set SYBOCS_DEBUG_LOGFILE=.\libsybfssl.log
C:\> isql –Usa –P –S my_ssl_server
C:\> type libsybfssl.log
```

jConnect support for Transact-SQL queries with COMPUTE clause

jConnect for JDBC now supports Transact-SQL® queries that include a COMPUTE clause. A COMPUTE clause allows you to display detail and summary results in one select statement. The summary row appears following the detail rows of a specific group. For example:

```
select type, price, advance
   from titles
   order by type
   compute sum(price), sum(advance) by type

type          price      advance
------------  ---------  ----------
UNDECIDED     NULL       NULL

Compute Result:
----------------------- ------------------------
NULL                    NULL
```

```
type          price      advance
------------  ---------  ----------
business      2.99       10,125.00
business      11.95      5,000.00

business      19.99      5,000.00
business      19.99      5,000.00

Compute Result:
----------------------- ------------------------
54.92                   25,125.00

...
...

(24 rows affected)
```

When jConnect executes a select statement that includes a COMPUTE clause, jConnect returns multiple result sets to the client, the number of result sets depends on the number of unique groupings available. Each group contains one result set for the detail rows and one result set for the summary. The client must process all result sets to fully process the rows returned; if it does not, only the detail rows of the first group of data are included in the first result set returned.

For more information about the COMPUTE clause, see the Adaptive Server Enterprise *Transact-SQL Users Guide*. For more information about processing multiple result sets, see the JDBC API documentation found on the Sun Microsystems Web site.

New feature in ESD #16

This section describes the new feature in ESD #16.

Support for Kazakh character set kz1048

Open Client and Open Server now support the character set kz1048. The character files included in this release are:

- *$SYBASE/charsets/kz1048/binary.srt*
- *$SYBASE/charsets/kz1048/charset.loc*

- *$SYBASE/charsets/unicode/kz1048.uct*

Note Adaptive Server needs to support the Kazakh character set before client applications can use Kazakh characters.

For information about internationalization and localization, including writing internationalized applications, see Open Client and Open Server *International Developer's Guide*.

New features in ESD #15

This section describes the new features in ESD #15.

SSL Plus 5.2.2 and SBGSE 2.2 support

Certicom Secure Sockets Layer (SSL) 5.2.2 support is provided for the Open Server and SDK components—Open Client (Client-Library), ESQL/C, and ESQL/COBOL—on the following platforms:

- HP Itanium HP-UX 11.23, 32-bit and 64-bit
- HP-UX 11.11, 32-bit and 64-bit
- IBM RS/6000 AIX 5.2, 32-bit and 64-bit
- Linux AMD64 RHEL 3.0, 64-bit
- Linux x86 RHEL 3.0, 32-bit
- Sun Solaris 8 (SPARC), 32-bit and 64-bit
- Sun Solaris 10 x64, 32-bit and 64-bit
- Windows x86 (2000 Service Pack 4 or later), 32-bit

Security Builder Government Solutions Edition (SBGSE) 2.2 is supported on the following SSL Plus 5.2.2 platforms:

- HP-UX 11.11, 32-bit and 64-bit
- IBM RS/6000 AIX 5.2, 32-bit and 64-bit
- Sun Solaris 8 (SPARC), 32-bit and 64-bit

- Windows x86(2000 Service Pack 4 or later), 32-bit

As a result of U.S. Federal regulations, Sybase is required to use cipher suites certified by the Federal Information Processing Standard (FIPS). After you upgrade to SDK 15.0 and to Open Server 15.0, perform the following:

- For Sun Solaris 8 (32-bit and 64-bit), HP-UX 11.11 (32-bit and 64-bit), and IBM AIX 5.2 (32-bit and 64-bit), be sure that the *lib3p* or *lib3p64* directory is in the dynamic load library path.
- In Windows, to allow the *ctlib* and *srvlib* libraries to find the Certicom cipher suites in *libsbgse2.dll*, add the following to the dynamic load library path:

  ```
  %SYBASE%\%SYBASE_OCS%\lib3p
  ```

  When running any application that uses SSL, be sure that the *lib3p* directory is in its dynamic load library path.

Open Server support for connection migration

The Open Server application programming interface (API) has been extended to enable Open Server to handle client connection migrations. The connection migration feature allows an Open Server application to dynamically distribute its load, provide transparent failover support, and, in cases where there are multiple Open Server applications that perform different functions, to redirect a client to an Open Server that can fulfill the client's request.

The new APIs enable Open Server to start, complete, and cancel a migration request, and to react to migration messages from the client. New event handlers and thread properties indicate client migration state changes. An Open Server can also detect whether a new connection is a migrating connection and retrieve a unique identifier from the connection.

The following sections discuss connection migration concepts, and the APIs that have been introduced or modified to support connection migration and their usage:

- In-batch migration and idle migration
- Context migration
- Modified Open Server API
- New Open Server APIs
- Error messages

- Instructing clients to migrate to a different server
- Accepting connections from migrated clients

Open Client migration is supported in ESD #8 and later versions. See “Open Client migration” on page 71 for more information.

For information about existing Open Server APIs, see the Open Server *Server-Library/C Reference Manual*.

In-batch migration and idle migration

With in-batch migration the client migrates while waiting for results from the original server. Conversely, with idle migration, the client is not waiting for any result from the original server.

In-batch migration enables Open Server to delay sending or completing results until after a connection has migrated. This is useful if Open Server cannot service the specific request or if it has no time to complete the request. With in-batch migration, Open Server can send a part of the result from the original server, and, after migration, the server the client has migrated to can send the other part of the result from the SRV_MIGRATE_RESUME event handler.

Note The original server can send a complete result to the client, in which case the new server does not send any result. Likewise, the original server may not send any result to the client, in which case, the new server must send the complete result to the client.

In an in-batch migration, your application must ensure that the unsent commands and messages are part of the client context. The new server must also access the number of rows affected by the command and the transaction state of the connection. The new server will send this information to the client using srv_senddone().

Context migration

Open Server supports seamless migration of the client’s connection. However, the responsibility of sharing and migrating the client’s context lies with your application. You can implement context migration in different ways such as through a shared file system or a network communication.

For an in-batch migration, the server that the client is migrating to does not know what type of event was raised in the original server. If your application needs this information, you must migrate the information as part of the client's context.

With idle migration, the client is not waiting for actual results from Open Server. Because there is no active query to migrate, idle migration is easier to implement than in-batch migration. However, idle migration still requires that your application fulfills any pending requests that may arrive before the client starts the migration.

Modified Open Server API

This section discusses the Open Server SRV_T_SESSIONID property which has been modified to support Open Server context migration. For more information about using SRV_T_SESSIONID in context migration, see "Instructing clients to migrate to a different server" on page 29.

SRV_T_SESSIONID property

The SRV_T_SESSIONID is a thread property that retrieves the session ID that the client sends to Open Server. As of ESD #15, an Open Server application can also set the SRV_T_SESSIONID property using the srv_thread_props() function, given that:

- The srv_thread_props(CS_SET, SRV_T_SESSIONID) call is made inside the SRV_CONNECT event handler and,
- The client supports connection migration or high availability.

This sample code sets the SRV_T_SESSIONID property:

```
CS_RETCODE ret;
CS_SESSIONID hasessionid;
ret = srv_thread_props(sp, CS_SET, SRV_T_SESSIONID,
   hasessionid, sizeof(hasessionid), NULL);
```

Note In versions earlier than ESD #14, for HA-failover, you must program an srv_negotiate() sequence to send the session ID to the client.

New Open Server APIs

This section discusses the new APIs that have been added to support connection migration. For more information about using these new APIs, see “Instructing clients to migrate to a different server” on page 29.

CS_REQ_MIGRATE

The CS_REQ_MIGRATE request capability indicates if a client supports the migration protocol and if the client is capable of migrating to another server when requested. You can use srv_capability_info() to retrieve the CS_REQ_MIGRATE capability information. For example:

```
CS_RETCODE ret;
CS_BOOL migratable;
ret = srv_capability_info(sp, CS_GET, CS_CAP_REQUEST,
   CS_REQ_MIGRATE, &migratable);
```

SRV_CTL_MIGRATE

SRV_CTL_MIGRATE has been added as a control type of the srv_send_ctlinfo() function. Using the SRV_CTL_MIGRATE control type, srv_send_ctlinfo() can now send a migration request to the client or cancel a previous migration request. SRV_CTL_MIGRATE can be used only if the client supports migration and has received a session ID when it first connected to the session.

Requesting a client migration

This sample code sends a request to the client to migrate to server “target”:

```
CS_RETCODE ret;
SRV_CTLITEM *srvitems;
CS_CHAR *target;
/*
** request a migration to server 'target'
*/
srvitems = (SRV_CTLITEM *) srv_alloc(sizeof
   (SRV_CTLITEM));
srvitems[0].srv_ctlitemtype = SRV_CT_SERVERNAME;
srvitems[0].srv_ctllength = strlen(target);
srvitems[0].srv_ctlptr = target;
ret = srv_send_ctlinfo(sp, SRV_CTL_MIGRATE, 1,
   srvitems);
srv_free(srvitems);
```

Your application can still send the SRV_CTL_MIGRATE control type even if a migration has already been requested. Open Server cancels the earlier migration request and sends a new request to the client. The return values for a new migration request are:

Return value	Description
CS_SUCCEED	The migration request was sent successfully.
CS_FAIL	The migration request failed due to one of the following reasons: • The Open Server thread does not support connection migration. • An earlier migration request was sent and the client has started migrating to the new server.

Cancelling a migration

You can also use the SRV_CTL_MIGRATE control type to cancel a previous migration request. In this case, *paramcnt* must be 0 and *param* must be a NULL pointer. For example:

```
ret = srv_send_ctlinfo(sp, SRV_CTL_MIGRATE, 0, NULL);
if (ret != CS_SUCCEED)
{
...
}
```

SRV_CTL_MIGRATE can be used by any thread in an Open Server application. However, a thread cancelling the migration of a client thread's connection has a different requirement than a client thread cancelling its own connection migration:

- Any Open Server thread can cancel a migration, however, the cancellation must be requested *before* the SRV_MIGRATE_STATE event handler informs the client thread that the client is ready to migrate.
- The client thread can cancel a migration even inside the SRV_MIGRATE_STATE event handler. However, the client cannot cancel a migration after it exits the SRV_MIGRATE_STATE event with a SRV_MIG_READY state.

The return values of a migration cancellation are:

Return value	Description
CS_SUCCEED	The migration request was cancelled successfully.
CS_FAIL	The migration cancellation failed due to one of these reasons: • There is no migration in progress. • The client has started migrating to the new server.

Note Open Server does not trigger a new migrate state event when a migration request is successfully cancelled.

SRV_MIGRATE_RESUME

When a client migrates to a new server while waiting for results, the new server invokes the SRV_MIGRATE_RESUME event after the client connection has successfully migrated. If the migration request failed or is cancelled, the event is invoked from the original server.

In the SRV_MIGRATE_RESUME event handler, your application does not have to send any actual result to the client, except for the SRV_DONE_FINAL result type that must *always* be sent. The only result that the default SRV_MIGRATE_RESUME sends to the client is SRV_DONE_FINAL.

This is an example of a SRV_MIGRATE_RESUME event handler:

```
/*
** Simple migrate_resume event handler.
*/
CS_RETCODE CS_PUBLIC
migrate_resume_handler(SRV_PROC *sp)
{
   CS_RETCODE ret;
   ret = srv_senddone(sp, SRV_DONE_FINAL,
      CS_TRAN_COMPLETED, 0);
   if (ret == CS_FAIL)
   {

      ...

   }
   return CS_SUCCEED;
}

...
```

```
/*
** Install the migrate-resume event handler
*/
srv_handle(server, SRV_MIGRATE_RESUME,
      migrate_resume_handler);
...
```

SRV_MIGRATE_STATE

SRV_MIGRATE_STATE is an event that is triggered whenever the migration state has transitioned to SRV_MIG_READY or SRV_MIG_FAILED, the transition being a result of a migration message from a client. The SRV_MIGRATE_STATE event handler is invoked in these situations:

SRV_T_MIGRATE_STATE	Situation	Possible application action
SRV_MIG_READY	The client has sent a message to the server indicating that it has detected the request and is ready to migrate. The server determines whether to continue the migration or not.	One of the following: • Make the context available for the other servers. • Cancel the migration if the application decides that migration is no longer needed. • Request another migration if a new migration target has been selected.
SRV_MIG_FAILED	The client has sent a message to the server indicating that the migration failed.	One of the following: • Access the client context and continue serving the connection. • Request another migration.

This is an example of a SRV_MIGRATE_STATE event handler:

```
/*
** Simple migrate-state event handler
*/
CS_RETCODE CS_PUBLIC
migrate_state_handler(SRV_PROC *sp)
{
   SRV_MIG_STATE migration_state;
   ret = srv_thread_props(sp, CS_GET,
      SRV_T_MIGRATE_STATE, &migration_state,
      sizeof (migration_state), NULL);

   ...

   switch(migration_state)
```

```
    {
       case SRV_MIG_READY:
            ...
       case SRV_MIG_FAILED:
            ...
    }
}

...

/*
** Install the migrate-state change event handler
*/
srv_handle(server, SRV_MIGRATE_STATE,
migrate_state_handler);
...
```

When working with the SRV_MIGRATE_STATE event handler:

- If the client thread cancels the migration from inside the SRV_MIGRATE_STATE event handler, your application must make sure that the context is consistent. For instance, you cannot expect a different server to use the context your application has created.
- If a new migration request is sent from within the SRV_MIGRATE_STATE event handler, this handler is called again when the client is ready to start with the new requested migration.

SRV_T_MIGRATE_STATE property and SRV_MIG_STATE enumerated type

SRV_T_MIGRATE_STATE indicates the migration state of the client. SRV_T_MIGRATE_STATE is a read-only property that any thread can access. The possible migration states are:

State	Value	Description
SRV_MIG_NONE	0	There is no migration in progress.
SRV_MIG_REQUESTED	1	A migration has been requested by the server.
SRV_MIG_READY	2	The client has received the request and is ready to migrate.
SRV_MIG_MIGRATING	3	The client is now migrating to the specified server.
SRV_MIG_CANCELLED	4	The migration request has been cancelled.
SRV_MIG_FAILED	5	The client failed to migrate.

SRV_MIG_STATE is an enumerated datatype that has been added to model the SRV_T_MIGRATE_STATE property. SRV_MIG_STATE is declared as:

```
typedef enum
```

```
{
   SRV_MIG_NONE,
   SRV_MIG_REQUESTED,
   SRV_MIG_READY,
   SRV_MIG_MIGRATING,
   SRV_MIG_CANCELLED,
   SRV_MIG_FAILED
} SRV_MIG_STATE;
```

This sample code shows how you can retrieve SRV_T_MIGRATE_STATE values; in case of a successful migration, the client exits and the SRV_DISCONNECT event handler is called with a SRV_MIG_MIGRATING status:

```
CS_RETCODE ret;
SRV_MIG_STATE migration_state;
ret = srv_thread_props(sp, CS_GET, SRV_T_MIGRATE_STATE,
   &migration_state, sizeof (migration_state), NULL);
if (ret != CS_SUCCEED)
{
...
}
```

SRV_T_MIGRATED

SRV_T_MIGRATED is a Boolean property that indicates whether a connection is a new connection or a migrated connection. This read-only property is set to true when the client is migrating or has migrated to the server. This sample code retrieves the value of SRV_T_MIGRATED:

```
CS_RETCODE ret;
CS_BOOL migrated;
status = srv_thread_props(sp, CS_GET, SRV_T_MIGRATED,
   &migrated, sizeof (migrated), NULL);
```

Error messages

These are the error messages that you might encounter when using the connection migration feature:

Error	Description
srv_thread_props(): Property - SRV_T_SESSIONID is not available	You try to retrieve a session ID that the client has not yet received.
srv_send_ctlinfo(SRV_CTL_MIGRATE): Connection cannot migrate	The client does not support migration.

Error	Description
srv_send_ctlinfo(SRV_CTL_MIGRATE): Migration can no longer be cancelled	You requested for a cancellation of a migration that has already started.
Migration failed but no SRV_MIGRATE_STATE handler was installed	The default SRV_MIGRATE_STATE handler detects a migration failure.

Instructing clients to migrate to a different server

This section discusses the requirements for an Open Server to migrate clients to other servers. When migrating clients to a different server your application must:

1. Create a unique session ID and send it to the clients in the connection handler.
2. Initiate connection migration.
3. Handle migration events.
4. Share the context of the connection, using the connection's session ID, to other servers.
5. (Optional) Act on ongoing migrations in existing handlers.

The following sections further discuss these activities.

Requesting a client to migrate

Open Server can use srv_send_ctlinfo() to send a migration request to the client. Client migration can be requested from any Open Server thread.

Managing the connect (SRV_CONNECT) event

In the SRV_CONNECT event handler, your application must:

- Check the SRV_T_MIGRATED property and determine if the connection is a migrated connection. If it is, your application must access the context based on the session ID provided by the client. The session ID can be retrieved using the SRV_T_SESSIONID thread property.
- Check CS_REQ_MIGRATE to determine if the client supports connection migration. If the client supports connection migration, your application must send a session ID using the SRV_T_SESSIONID property to the client if the client has not yet received a session ID. By assigning the client a session ID, your application can instruct the client to migrate when the need arises.

Managing the migrate state (SRV_MIGRATE_STATE) event

The SRV_MIGRATE_STATE event handler must manage the migration state changes and execute the actions appropriate for each change:

- SRV_MIGRATE_STATE changed to SRV_MIG_READY

  A “ready” migration state indicates that the client is prepared to migrate and, for now, is not going to send any request. In the SRV_MIGRATE_STATE event handler, Open Server shares the client context with the server the client is migrating to. Afterwards, your application can return from the event handler, and Open Server can automatically instruct the client to start the migration.

- SRV_MIGRATE_STATE changed to SRV_MIG_FAILED

  If the SRV_MIGRATE_STATE event handler is triggered because the migration state changed to “failed,” your application must access the context again. Your application can request another migration attempt from the SRV_MIG_STATE event handler using the srv_send_ctlinfo() function. However, the client may have sent another query before it indicates it is ready to migrate again. The application must be able to service or migrate such a request.

Sharing client context

For servers to start and continue servicing a client, the servers must have access to the client’s context which is identified by the client’s session ID. Typically, the client’s context contains data, such as global data, that event handlers for the client can access. The amount of context required for a connection depends on the service that the Open Server application provides. The more context-free the service is, the less context needs to be shared.

Managing the migrate resume (SRV_MIGRATE_RESUME) event

Your application sends the remaining results and messages to the client inside the SRV_MIGRATE_RESUME event handler. The results and messages that Open Server sends to the client depend on your application and the migration type. However, your application must end the SRV_MIGRATE_RESUME event handler by sending the SRV_DONE_FINAL result type to the client.

Managing the disconnect (SRV_DISCONNECT) event

In the SRV_DISCONNECT event handler, your application must check SRV_T_MIGRATE_STATE to determine the client’s migration state:

- A migration state of SRV_MIG_REQUESTED indicates that the SRV_DISCONNECT event has been triggered because the Open Server application terminated the connection before the client could respond to the migration request.
- A migration state of SRV_MIG_MIGRATING indicates that the SRV_DISCONNECT event has been triggered because the client application, after a successful migrating to the new server, closed the connection.
- For all other migration states, the client must make sure that connection-specific context is cleaned up because no other server will pick up this context.

Managing in-batch migration

An event handler that runs for long periods of time must occasionally inspect the migration state. Other Open Server threads can send a migration request even while an event handler process is still running. In this case, the event handler, if it is able to, must interrupt the process, and postpone the generation and sending of results until the connection has migrated to the new server.

Attention handling

When a client sends an attention message to cancel an outstanding request, the SRV_T_GOTATTENTION thread property is set to CS_TRUE and the SRV_ATTENTION event handler is called. The specific attention handling needs of a connection migration are described below:

- For the SRV_MIGRATE_STATE event handler and SRV_MIG_READY state:

  If the attention message arrives in the SRV_MIGRATE_STATE event handler before the client indicates that it is ready to migrate, Open Server acknowledges the attention when the SRV_MIGRATE_STATE event handler ends. This completes the request from the client. After a successful migration, the server that the client has migrated to does not receive this attention message and, because the client is not waiting for results from Open Server, the SRV_MIGRATE_RESUME event handler is not called.

  Thus, your application must check if the SRV_T_GOTATTENTION property is set to CS_TRUE before making the context available to other servers. If SRV_T_GOTATTENTION is set to CS_TRUE, you must update the context to indicate that the client has cancelled the operation.

- For the SRV_MIGRATE_RESUME event handler:

  If the client has sent the attention message after the client indicated that it is ready to migrate and the migration succeeded, the attention is sent to the server to which the client has migrated. It is therefore possible that, after a successful migration, an attention can be received by the SRV_MIGRATE_RESUME event handler even if the original server has updated the context to reflect the cancellation. Thus, your application must check if the client has sent an attention to the server before it can execute the SRV_MIGRATE_RESUME event handler.

Disconnecting Open Server

Your application can terminate a client connection even when a migration has been requested; however, a new client command that is sent just before Open Server issued the termination command may get lost. To avoid this, your application must:

- If possible, avoid terminating connections when a client is instructed to migrate.
- If there is a need to disconnect a client, Open Server must set a reasonable wait time before requesting the migration. This gives a client the time to detect the migration request before it issues another command.
- When Open Server terminates a connection, the SRV_DISCONNECT event handler is called. Inside this handler, ensure that the context is available to other servers if the migration state is still set to SRV_MIG_REQUESTED.

Accepting connections from migrated clients

Open Server can determine if a new connection is migrating or has migrated by inspecting the SRV_T_MIGRATED property in the SRV_CONNECT event handler. If SRV_T_MIGRATED is TRUE, you can retrieve the session ID from the client using the SRV_T_SESSIONID property. You can also change the session ID, but this is not required to migrate the client later.

If the client was executing a command when it migrated, the SRV_MIGRATE_RESUME event is triggered and Open Server can send results to the client to complete the command. Your application is responsible for retrieving the session information. You must also determine whether you still need to send results to the client from within the SRV_MIGRATE_RESUME event handler.

Support for partial update of text and image columns

Open Client and Open Server now support the partial update of text and image columns. A partial update allows you to specify the part of the text or image field that you want to replace, delete, or insert at, and update that part only instead of modifying the entire field. For more information about text and image data handling, see the Open Client *Client-Library /C Reference Manual*.

Note Currently, ASE does not support partial update of text or image columns.

Client-Library usage

This section discusses the Open Client APIs that have been introduced or modified to support partial updates. For a sample program, see *$SYBASE/$SYBASE_OCS/sample/ctlib/uctext.c*.

CS_PARTIAL_TEXT

CS_PARTIAL_TEXT is a new property that indicates whether or not the client needs to perform a partial update. You can set this property in the connection or context level using ct_con_props() or ct_config(), respectively. The possible values of CS_PARTIAL_TEXT are CS_TRUE and CS_FALSE.

The CS_PARTIAL_TEXT property must be set before a connection to the server is established. If the server does not support partial updates, CS_PARTIAL_TEXT is set to CS_FALSE, which is the default value.

CS_IODESC

The CS_IODESC structure has been modified to support the partial update of text and image columns. The new *iotype* value CS_IOPARTIAL specifies that a partial update must be performed on the text or image column while the new field *delete_length* indicates the number of bytes that must be overwritten or deleted from the column. The reserved field *offset*, on the other hand, indicates the first byte in the column that is affected by the partial update

To perform a partial update, use ct_data_info() to set *iotype*, *delete_length*, and *offset*. The values of *delete_length* and of the data passed to the server through ct_send_data() determine the behavior of the partial update:

delete_length	Text data	Server action
0	Provided	Insert the text data at *offset*.

delete_length	Text data	Server action
!= 0	Provided	Starting from *offset*, overwrite *delete_length* bytes of data with text data.
!= 0	Not provided	Starting at *offset*, delete *delete_length* bytes of data.
NULL	Provided / Not provided	Delete data starting from *offset* to the end of the text or image column.

New CS_IODESC structure

The new CS_IODESC structure is:

```
typedef struct _cs_iodesc
{
        CS_INT iotype;
        CS_INT datatype;
        CS_LOCALE *locale;
        CS_INT usertype;
        CS_INT total_txtlen;
        CS_INT offset;
        CS_BOOL log_on_update;
        CS_CHAR name[CS_OBJ_NAME];
        CS_INT namelen;
        CS_BYTE timestamp[CS_TS_SIZE];
        CS_INT timestamplen;
        CS_BYTE textptr[CS_TP_SIZE];
        CS_INT textptrlen;
        CS_INT delete_length;
} CS_IODESC;
```

Backward compatibility

Applications that have been compiled with the old CS_IODESC structure definition can still link to shared libraries that use the new CS_IODESC structure. Because the old application has not set the *iotype* to CS_IOPARTIAL, the new shared libraries will not access *delete_length*, thus maintaining backward compatibility.

ct_send_data()

ct_send_data() has been extended to support partial updates. Currently, to send data, ct_send_data() constructs a Transact-SQL writetext statement and the data is sent in chunks using multiple ct_send_data() calls. You can use this same method to send partially updated data. However, ct_send_data() will construct an udpatetext statement instead of a writetext. The updatetext syntax is:

```
updatetext table_name.column_name text_pointer
```

```
{NULL | offset} {NULL | delete_length} [with_log]
```

Note The updatetext statement is created only if *iotype* is set to CS_IOPARTIAL.

This example shows ct_send_data() sending partial update data:

```
/*
** UpdateTextData()
*/
CS_STATIC CS_RETCODE
UpdateTextData(connection, textdata, newdata)
CS_CONNECTION connection;
TEXT_DATA textdata;
char *newdata;
{
   CS_RETCODE retcode;
   CS_INT res_type;
   CS_COMMAND  *cmd;
   CS_INT      i;
   CS_TEXT     *txtptr;
   CS_INT       txtlen;
   /*
   ** Allocate a command handle to send the text with
   */
   ...CODE DELETED.....
   /*
   ** Inform Client-Library the next data sent will
   ** be used for a text or image update.
   */
   if ((retcode = ct_command(cmd, CS_SEND_DATA_CMD,
   NULL, CS_UNUSED, CS_COLUMN_DATA)) != CS_SUCCEED)
   {
      ex_error("UpdateTextData: ct_command() \
         failed");
      return retcode;
   }
   /*
   ** Fill in the description information for the
   ** update and send it to Client-Library.
   */
   txtptr = (CS_TEXT *)newdata;
   txtlen = strlen(newdata);
   textdata->iodesc.total_txtlen = txtlen;
   textdata->iodesc.log_on_update = CS_TRUE;
   /*
```

```
**  Insert newdata at offset 20.
*/
textdata->iodesc.iotype = CS_IOPARTIAL;
textdata->iodesc.offset = 20;
textdata->iodesc.delete_length = 0;
retcode = ct_data_info(cmd, CS_SET, CS_UNUSED,
&textdata->iodesc);
if (retcode != CS_SUCCEED)
{
   ex_error("UpdateTextData: ct_data_info() \
      failed");
   return retcode;
}
/*
** Send the text one byte at a time. This is not
** the best thing to do for performance reasons,
** but does demonstrate that ct_send_data()
** can handle arbitrary amounts of data.
*/
for (i = 0; i < txtlen; i++, txtptr++)
{
   retcode = ct_send_data(cmd, txtptr,(CS_INT)1);
   if (retcode != CS_SUCCEED)
   {
      ex_error("UpdateTextData: ct_send_data() \
         failed");
      return retcode;
   }
}
/*
** ct_send_data() writes to internal network
** buffers. To insure that all the data is
** flushed to the server, a ct_send() is done.
*/
if ((retcode = ct_send(cmd)) != CS_SUCCEED)
{
   ex_error("UpdateTextData: ct_send() failed");
   return retcode;
}
/* Process the results of the command */
while ((retcode = ct_results(cmd, &res_type)) ==
   CS_SUCCEED)
{
   switch ((int)res_type)
   {
      case CS_PARAM_RESULT:
```

```
                    /*
                    ** Retrieve a description of the
                    ** parameter data. Only timestamp data is
                    ** expected in this example.
                    */
                    retcode = ProcessTimestamp(cmd, textdata);
                    if (retcode != CS_SUCCEED)
                    {
                        ex_error("UpdateTextData: \
                          ProcessTimestamp() failed");
                        /*
                        ** Something failed, so cancel all
                        ** results.
                        */
                        ct_cancel(NULL, cmd, CS_CANCEL_ALL);
                        return retcode;
                    }
                    break;
                case CS_CMD_SUCCEED:
                case CS_CMD_DONE:
                    /*
                    ** This means that the command succeeded
                    ** or is finished.
                    */
                    break;
                case CS_CMD_FAIL:
                    /*
                    ** The server encountered an error while
                    ** processing our command.
                    */
                    ex_error("UpdateTextData: ct_results() \
                        returned CS_CMD_FAIL");
                    break;
                default:
                    /*
                    ** We got something unexpected.
                    */
                    ex_error("UpdateTextData: ct_results() \
                        returned unexpected result type”);
                    /* Cancel all results */
                    ct_cancel(NULL, cmd, CS_CANCEL_ALL);
                    break;
            }
        }
        /*
        ** We're done processing results. Let's check the
```

```
        ** return value of ct_results() to see if
        ** everything went ok.
        */
        ...CODE DELETED.....
        return retcode;
    }
```

For information about ct_send_data(), see the Open Client *Client-Library/C Reference Manual*.

Handling of unitext data

If the client application uses 2-byte Unicode datatypes, the application must make sure that it sends an even number of bytes to avoid a character split. You can use the *buflen* parameter of ct_send_data() and the *total_txtlen* field of CS_IODESC to specify the length, in bytes, of the Unicode data. For Unitext, the *offset* and *delete_length* values must be specified as a character count while *total_txtlen* must be specified in bytes. For other datatypes, the *offset*, *delete_length*, and *total_txtlen* must be in bytes.

Open Server usage

This section discusses how Open Server must be set up to support partial updates.

sp_mda

sp_mda is a stored procedure that retrieves metadata from the server. To support partial updates, your Open Server application must define an sp_mda stored procedure and specify the updatetext syntax that an Open Client application must use.

An Open Client application must invoke sp_mda using these parameters and values:

Parameter	Value	Description
clienttype	5	5 indicates that the client is Client-Library.
mdaversion	1	
clientversion	0	clientversion is an optional parameter that indicates the client version. The default is 0.

If the server supports partial updates, sp_mda returns:

Parameter	Value
mdinfo	“UPDATETEXT”

Parameter	Value
querytype	2
query	*updatetext_syntax* Example: `updatetext ? ? ? {NULL | ?} {NULL | ?}` where "?" indicates the updatetext parameters.

For more information about the sp_mda stored procedure, see the Mainframe Connect™ DB2 UDB Options for IBM CICS and IMS *Installation and Administration Guide*. For a sample implementation of sp_mda, see *$SYBASE/$SYBASE_OCS/sample/srvlibrary/updtext.c*.

SRV_T_BULKTYPE

To correctly retrieve the partially updated data sent by the client, the Open Server application must set SRV_T_BULKTYPE to SRV_TEXTLOAD, SRV_UNITEXTLOAD, or SRV_IMAGELOAD. For more information about SRV_T_BULKTYPE, see the Open Server *Server-Library/C Reference Manual*.

Handlers

The SRV_LANGUAGE and SRV_BULK handlers have to be installed in Open Server. Open Server uses SRV_LANGUAGE to receive the updatetext statement from the Client-Library. SRV_BULK, on the other hand, receives the data sent through ct_send_data().

For more information about SRV_LANGUAGE and SRV_BULK, see the Open Client and Open Server *Common Libraries Reference Manual* and the Open Server *Server-Library/C Reference Manual*.

New features in ESD #14

This section describes the new features in ESD #14.

LDAP schema for Microsoft Active Directory

The Lightweight Directory Access Protocol (LDAP) is an industry standard for accessing directory services. Directory services allow applications to look up information using a distinguished name (DN) from an LDAP server that stores and manages user, server, and software information that is used throughout the enterprise or over a network. Currently, Sybase supports LDAP directory schema to store, modify, and retrieve server connection information for an OpenLDAP directory service and for a Netscape directory service.

As of ESD #14, Sybase also supports LDAP for Microsoft Active Directory. You can find *sybase.ldf*, a new directory schema in Unicode format, in these locations:

- For Windows:

  %SYBASE%\%SYBASE_OCS%\ini

- For UNIX:

  $SYBASE/$SYBASE_OCS/config

Importing the directory schema

You can import *sybase.ldf* into the Active Directory (AD) or into an Active Directory Application Mode (ADAM) instance using the ldifde.exe command provided in the ADAM installation. To import the directory schema, execute the ldifde.exe command from the ADAM installation using this syntax:

```
ldifde -i -u -f sybase.ldf -s server:port -b username
domain password -j . -c "cn=Configuration,dc=X"
#configurationNamingContext
```

Creating a container for Sybase server entries

After the schema has been successfully imported into the Active Directory, you can create a container for the Sybase server entries and set appropriate read and write permissions for the container and its child objects.

Example A container with a relative distinguished name (RDN) “CN=SybaseServers” is created in the root of the Active Directory for domain “mycompany.com” to store and retrieve Sybase server entries. The root distinguished name (rootDN) for this container is reflected in the *libtcl.cfg* file as:

```
ldap=libsybdldap.dll ldap://localhost:389/
cn=SybaseServers,dc=mycompany,dc=com??...
```

If you create a dedicated user account name “Manager” with password “secret” in the Active Directory to add and modify Sybase server entries, the complete entry in the *libtcl.cfg* file is:

- For Windows:

```
ldap=libsybdldap.dll
ldap://localhost:389/cn=SybaseServers,dc=mycompany,
dc=com????bindname=cn=Manager,cn=Users,dc=mycompay,
dc=com?secret
```

- For UNIX:

```
ldap=libsybdldap.so
ldap://myADhost:389/cn=SybaseServers,dc=mycompany,
dc=com????bindname=cn=Manager,cn=Users,dc=mycompay,
dc=com?secret
```

After setting the appropriate read and write permissions, you will be able to use the Sybase utility programs such as dscp or dsedit to store, view, and modify Sybase server entries in the Active Directory.

For more information about extending an Active Directory schema, search for “Extending the Schema” on the Microsoft Web site.

Support for ESQL/C and ESQL/COBOL 64-bit applications

ESD #14 includes the 64-bit precompilers for ESQL/C and ESQL/COBOL. You can use these precompilers to build 64-bit applications on:

- For ESQL/C – all 64-bit platforms
- For ESQL/COBOL – Sun Solaris 8 (SPARC) 64-bit, HP-UX 64-bit, HP Itanium 64-bit, and IBM AIX 64-bit

This table lists the non-reentrant and reentrant versions of the 64-bit precompilers for ESQL/C and ESQL/COBOL:

ESQL host language	Non-reentrant version	Reentrant version
ESQL/C	cpre64	cpre_r64
ESQL/COBOL	cobpre64	cobpre_r64

Compiler build mode

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 must be set to 32 for a 32-bit build, and to 64 for a 64-bit build. Failure to do this can result to a build error or produce an unexpected binary.

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 your 64-bit application, see the sybopts.sh script available in the *$SYBASE/$SYBASE_OCS/sample/esqlc* and the *$SYBASE/$SYBASE_OCS/sample/esqlcob* directories. See also the Open Client and Open Server *Programmer's Supplement* for your platform.

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, a short 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.

New ESQL/COBOL veneer layer libraries

Shared dynamic ESQL/COBOL veneer layer libraries are introduced in ESD #14 and are released on all 32-bit and 64-bit platforms that support ESQL/Cobol:

Plaform	Library name	Reentrant version
HP-UX PA-RISC 32-bit	*libsybcobct.sl*	*libsybcobct_r.sl*
HP-UX PA-RISC 64-bit	*libsybcobct64.sl*	*libsybcobct_r64.sl*
All other 32-bit platforms that support ESQL/COBOL	*libsybcobct.so*	*libsybcobct_r.so*
All other 64-bit platforms that support ESQL/COBOL	*libsybcobct64.so*	*libsybcobct_r64.so*

The existing static version of the ESQL/COBOL veneer layer library is called *libsybcobct.a*.

New font option for Open Server and SDK installers

When installing in graphic user interface (GUI) mode, you can specify the installer display font by using the -font command line option or the *font.ini* file. This option is useful when your Java virtual machine (JVM) is unable to correctly pick the font to display for your locale, and the product installers display garbage characters.

Note If you use both the -font command line option and the *font.ini* file, the font specified using the -font command line option takes precedence.

Using the *-font* command line option

You can modify the default installer display font when you start the installer. For example:

```
./setup -is:javaconsole -font "FZFangSong"
```

The example uses the Chinese font FZFangSong and requires that the LANG environment variable is set to a Chinese locale such as zh_CN.

To ensure that characters are displayed correctly, the JAVA_FONTS environment variable must point to the location of the font files. Otherwise, the installer terminates with this message:

```
Invalid command line option: unable to find fontname
font.
Make sure the font name is correct and JAVA_FONTS
environment variable is set.
```

Using the *font.ini* file

Another way to specify the installer display font is to create a *font.ini* file in the installer image root directory. The *font.ini* file specifies the font name and the font location. For example:

```
#Set to Kochi Mincho font
font=Kochi Mincho
path=/usr/share/fonts/ja/TrueType
```

The example uses the Japanese font Kochi Mincho and requires that the LANG environment variable is set to a Japanese locale such as ja_JP. If the font or the font path specified in the *font.ini* file is incorrect, the installer terminates with this message:

```
Error: Unable to find fontname font.
Make sure font name and path are correct in font.ini
file.
```

Extended platform support for the ASE ODBC Driver by Sybase

A 64-bit version of the ASE ODBC Driver is now supported on Linux x86-64 platform. The ASE ODBC 64-bit driver is compatible with the unixODBC driver manager and is installed in the *$SYBASE/DataAccess64/ODBC* directory.

New jConnect scripts to access SQL Anywhere metadata

To support JDBC DatabaseMetaData methods, Sybase provides a set of stored procedures that jConnect can call for metadata about a database. These stored procedures must be installed on the server for the JDBC metadata methods to work.

Previously, jConnect provided only one script to install the SQL Anywhere® stored procedures. In ESD #14, jConnect separates the metadata script files for the different versions of SQL Anywhere:

- *sql_asa.sql* – installs stored procedures on the SQL Anywhere 9.x database
- *sql_asa10.sql* – installs stored procedures on the SQL Anywhere 10.x database
- *sql_asa11.sql* – installs stored procedures on the SQL Anywhere 11.x database

For more information, see the "Accessing database metadata" section of the jConnect for JDBC *Programmer's Reference*.

Enhancement to the jConnect extended password encryption

Starting in ESD #14, you can use the Certicom Security Builder GSE-J to perform RSA encryption of password. Certicom Security Builder GSE-J is a FIPS 140-2 compliant JCE provider that is included in your copy of the jConnect for JDBC Driver. This provider contains two jar files, *EccpressoFIPS.jar* and *EccpressoFIPSJca.jar*, that are both accessible from the *$JDBC_HOME/classes* and the *$JDBC_HOME/devclasses* directories.

To use the Certicom Security Builder GSE-J provider, you must set the value of JCE_PROVIDER_CLASS connection property to "com.certicom.ecc.jcae.Certicom" and add the *EccpressoFIPS.jar* and *EccpressoFIPSJca.jar* files to the CLASSPATH. See the "Setting up the Java Cryptography Extension (JCE) provider" on page 75 for details.

Note If you enable password encryption by setting the ENCRYPT_PASSWORD connection property but not the JCE_PROVIDER_CLASS connection property, jConnect attempts to locate and load the Certicom Security Builder GSE-J provider. This will succeed only if *EccpressoFIPS.jar* and *EccpressoFIPSJca.jar* are located in the same directory as the jConnect jar file—*jconn3.jar* or *jconn3d.jar*— in use.

New feature in ESD #13

This section describes the new feature in ESD #13.

isql support for obfuscated input

With the new --conceal command line option, you can hide your input during an isql session. The --conceal option is useful when entering sensitive information, such as passwords. The syntax for the --conceal option is:

```
isql --conceal [':?' | 'wildcard']
```

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.

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.

Examples

Example 1 Changes password 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 2 Changes password 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 3 Activates a role for the current user. This example 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>
```

New features in ESD #12

This section describes features that are new for ESD #12.

New *cs_config* properties

Three properties are added to the cs_config routine to specify an alternative location specific to the property. These properties are CS_SYBASE_HOME, CS_LIBTCL_CFG and CS_DEFAULT_IFILE.

CS_SYBASE_HOME

The CS_SYBASE_HOME property specifies the name and path to an alternate Sybase home directory and overrides the environment variable *$SYBASE* (*%SYBASE%* for Windows).

CS_SYBASE_HOME must be set before allocating a CS-Library context because the allocation of a context requires a valid Sybase home directory from which it will be set up. This means that CS_SYBASE_HOME must be set *before* calling cs_ctx_alloc() or cs_ctx_global(). cs_config() has to be invoked with a NULL context to set CS_SYBASE_HOME.

Example

```
ret = cs_config(NULL, CS_SET, CS_SYBASE_HOME,
"/work/NewSybase", CS_NULLTERM, NULL);
```

You can also use the isql and bcp utilities to specify an alternate Sybase home directory. For details, see "isql and bcp enhancement" on page 51.

CS_LIBTCL_CFG

The CS_LIBTCL_CFG property specifies the name and path to an alternate *libtcl.cfg* file. As in the CS_SYBASE_HOME property, CS_LIBTCL_CFG is set by cs_config() using a NULL context and must be set before a CS-Library context is allocated.

Example

```
ret = cs_config(NULL, CS_SET, CS_LIBTCL_CFG,
"/work/Sybase/OCS-15_0/config/libtcl.cfg",
CS_NULLTERM, NULL);
```

CS_DEFAULT_IFILE

The CS_DEFAULT_IFILE property specifies the name of the alternate *interfaces* file and its path. Unlike the CT-Library property CS_IFILE, CS_DEFAULT_IFILE does not override the use of alternate directory services that have already been specified in the *libtcl.cfg* file. The primary purpose of CS_DEFAULT_IFILE is to set a new default location for the *interfaces* file, in case the *interfaces* file is being used as the directory service.

A CS-Library context must be allocated before calling cs_config() and it must be passed in cs_config() while setting the CS_DEFAULT_IFILE property.

Example

```
ret = cs_config(ctx, CS_SET, CS_DEFAULT_IFILE,
"/work/NewSybase/interfaces", CS_NULLTERM, NULL);
```

isql and *bcp* enhancement

You can now set an alternate Sybase home directory using the new isql and bcp option -y.

Examples **Example 1** Sets an alternate Sybase home directory in isql:

```
isql -y/work/NewSybase -Uuser1 -Psecret -SMYSERVER
```

Example 2 Sets an alternate Sybase home directory in bcp:

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

isql support for dynamic redirection of T-SQL output

You can now redirect the output of a T-SQL command to a file or pipe it to an external application from within an interactive isql session.

Examples

Example 1 Writes the output of the select @@servername command to the file *myserver.txt*, or overwrites that file if it already exists:

```
1> select @@servername
2> go > myserver.txt
```

Example 2 Writes the output of the select @@version command to the new file *myserver.txt*, or appends it to that file if it already exists:

```
1> select @@version
2> go >> myserver.txt
```

Example 3 Pipes the output of the sp_who command to grep and returns the lines that contain the string 'sa':

```
1> sp_who
2> go | grep sa
```

jConnect, ODBC, OLEDB and ADO.NET support for Adaptive Server Cluster Edition

This section describes the ASE driver features that support the Cluster Edition environment.

Login redirection

At any given time, some servers within a Cluster Edition environment are usually more loaded with work than others. When a client application attempts to connect to a busy server, the login redirection feature helps balance the load of the servers by allowing the server to redirect the client connection to less busy servers within the cluster. The login redirection occurs during the login sequence and the client application does not receive notification that it was redirected.

Note When a client application connects to a server that is configured to redirect clients, the login time may increase because the login process is restarted whenever a client connection is redirected to another server.

Connection migration

The connection migration feature allows a server in a Cluster Edition environment to dynamically distribute load, and seamlessly migrate an existing client connection and its context to another server within the cluster. This feature enables the Cluster Edition environment to achieve optimal resource utilization and decrease computing time. Because migration between servers is seamless, the connection migration feature also helps create a truly High Availability (HA), zero-downtime environment.

Note The login redirection and connection migration features are enabled automatically when a client application connects to a server that supports these features.

Note Command execution time may increase during server migration. Sybase recommends that you increase the command timeouts accordingly.

Connection failover enhancement

Existing connection failover allows a client application to switch to an alternate ASE server if the primary server becomes unavailable due to an unplanned event, like power outage or a socket failure. This feature is enhanced to allow client applications to failover numerous times to multiple servers using dynamic failover addresses.

With the High Availability enabled, the client application does not need to be configured to know the possible failover targets. ASE keeps the client updated with the best failover list based on cluster membership, logical cluster usage and load distribution. During failover, the client refers to the ordered failover list while attempting to reconnect. If the driver successfully connects to a server, the driver internally updates the list of host values based on the list returned. Otherwise, the driver throws a connection failure exception.

Enabling extended connection failover in jConnect

You can use the connection string to enable connection failover by setting REQUEST_HA_SESSION to true. For example:

```
URL="jdbc:sybase:Tds:server1:port1,server2:port2,...,
serverN:portN/mydb?REQUEST_HA_SESSION=true"
```

where server1:port1, server2:port2, ... , serverN:portN is the ordered failover list.

In establishing a connection, jConnect will try to connect to the first host and port specified in the failover list. If unsuccessful, it will go through the list until a connection is established or until the end of the list is reached.

Note The list of alternate servers specified in the connection string is used only during initial connection. After the connection is established with any available instance, and the client supports high-availability, the client will receive an updated list of the best possible failover targets from the server. This new list overrides the specified list.

Enabling extended connection failover in ODBC

Using the ODBC user interface

One way to enable the extended connection failover in ODBC is through the ODBC user interface.

❖ **Using the user interface to enable extended failover**

1 Open the Adaptive Server Enterprise dialog box.

2 Go to the Connection tab.

3 Select Enable High Availability.

4 Optionally, enter the alternate servers and ports in the Alternate Servers field using the format:

```
server1:port1,server2:port2,...,serverN:portN;
```

In establishing a connection, ODBC Driver by Sybase will first try to connect to the primary host and port defined in the General tab of the Adaptive Server Enterprise dialog box. If ODBC fails to establish a connection, ODBC will go through the list of hosts and ports specified in the Alternate Servers field.

Using the ODBC connection string

Another way to enable the connection failover in ODBC is to set the HASession connection string property to 1. You can use SQLDriverConnect to specify a connection string. For example:

```
Driver=AdaptiveServerEnterprise;server=server1;
port=port1;UID=sa;PWD=;HASession=1;
AlternateServers=server2:port2,...,serverN:portN;
```

The preceding example defines server1 and port1 as the primary server and port. If ODBC fails to establish connection to the primary server, and alternate servers are defined, it will go through the ordered list of servers and ports specified in the Alternate Servers field until a connection is established or until the end of the list is reached.

Note The list of alternate servers specified in the GUI or the connection string is used only during initial connection. After the connection is established with any available instance, and the client supports high-availability, the client will receive an updated list of the best possible failover targets from the server. This new list overrides the specified list.

Enabling extended connection failover in OLEDB

Using the OLEDB user interface

One way to enable the extended connection failover in OLEDB is through the OLEDB user interface.

❖ **Using the user interface to enable extended failover**

1. Open the Configure Data Source dialog box of the Sybase Datasource Administrator.
2. Go to the Connection tab.
3. Select Enable High Availability.
4. Optionally, enter the alternate servers and ports in the Alternate Servers field using the format:

   ```
   server1:port1,server2:port2,...,serverN:portN;
   ```

In establishing a connection, OLEDB Provider by Sybase will first try to connect to the primary host and port defined in the General tab of the Configure Data Source dialog box. If OLEDB fails to establish a connection, OLEDB will go through the list of hosts and ports specified in the Alternate Servers field.

Using the OLEDB connection string

You can also use the OLEDB connection string to enable extended failover by setting the HASession connection string property to 1. For example:

```
Provider=ASEOLEDB;User ID=sa;Password=;
InitialCatalog=sdc;Data Source=server1:port1;
ProviderString='HASession=1;AlternateServers=
server2:port2,...,serverN:portN';
```

In the preceding example, Data Source defines the primary server and port. OLEDB Provider by Sybase will try to establish connection to the primary server first. If unsuccessful, and alternate servers are defined, OLEDB will go through the servers listed in the Alternate Servers field until a connection is established or until the end of the list is reached.

Note The list of alternate servers specified in the GUI or the connection string is used only during initial connection. After the connection is established with any available instance and the client supports high-availability, the client will receive an updated list of the best possible failover targets from the server. This new list overrides the specified list.

Enabling extended connection failover in ADO.NET

To enable the extended failover, you need to set the HASession connection string property to 1. For example:

```
Data Source=server1;Port=port1;User ID=sa;Password=;
Initial Catalog=sdc;HASession=1;
AlternateServers=server2:port2,...,serverN:portN;
```

In the preceding example, Data Source defines the primary server and port. ADO.NET Provider by Sybase will try to establish connection to the primary server first and, if unsuccessful, will go through the servers listed in Alternate Servers until a connection is established or until the end of the list is reached.

Note The list of alternate servers specified in the connection string is used only during initial connection. After the connection is established with any available instance, and the client supports high-availability, the client will receive an updated list of the best possible failover targets from the server. This new list overrides the specified list.

New feature in ESD #11

This section describes the new feature in ESD #11.

SSL enhancement for common name validation

The default behavior for SSL validation in Open Client and Open Server is to compare the common name in the server's certificate with the server name specified by ct_connect(). In a Shared Disk Cluster (SDC) environment, a client may specify the SSL certificate common name independent of the server name or the SDC instance name. A client may connect to an SDC by its cluster name—which represents multiple server instances—or to a specific server instance.

With the SSL enhancement, Open Client and Open Server can now support common name validation in an SDC environment. This enhancement allows the ASE SSL certificate common name to be different from the server or cluster name by allowing the client to use the transport address to specify the common name used in the certificate validation. The transport address can be specified in one of the directory services like the *interfaces* file, LDAP or NT registry, or through the connection property CS_SERVERADDR.

New syntax for UNIX

This is the new syntax of the server entries for the SSL-enabled ASE and cluster for UNIX:

```
CLUSTERSSL
query tcp ether hostname1 5000 ssl="CN=name1"
query tcp ether hostname2 5000 ssl="CN=name2"
```

```
query tcp ether hostname3 5000 ssl="CN=name3"
query tcp ether hostname4 5000 ssl="CN=name4"

ASESSL1
master tcp ether hostname1 5000 ssl="CN=name1"
query tcp ether hostname1 5000 ssl="CN=name1"

ASESSL2
master tcp ether hostname2 5000 ssl="CN=name2"
query tcp ether hostname2 5000 ssl="CN=name2"

ASESSL3
master tcp ether hostname3 5000 ssl="CN=name3"
query tcp ether hostname3 5000 ssl="CN=name3"

ASESSL4
master tcp ether hostname1 5000 ssl="CN=name4"
query tcp ether hostname1 5000 ssl="CN=name4"
```

New syntax for Windows

This is the new syntax of the server entries for the SSL-enabled ASE and cluster for Windows:

```
[CLUSTERSSL]
query=tcp,hostname1,5000, ssl="CN=name1"
query=tcp,hostname2,5000, ssl="CN=name2"
query=tcp,hostname3,5000, ssl="CN=name3"
query=tcp,hostname4,5000, ssl="CN=name4"

[ASESSL1]
master=tcp,hostname1,5000, ssl="CN=name1"
query=tcp,hostname1,5000, ssl="CN=name1"

[ASESSL2]
master=tcp,hostname2,5000, ssl="CN=name2"
query=tcp,hostname2,5000, ssl="CN=name2"

[ASESSL3]
master=tcp,hostname3,5000, ssl="CN=name3"
query=tcp,hostname3,5000, ssl"CN=name3"

[ASESSL4]
master=tcp,hostname4,5000, ssl="CN=name4"
query=tcp,hostname4,5000, ssl="CN=name4"
```

New features in ESD #10

This section describes features that are new for ESD #10.

Kerberos support using Windows SSPI

This feature provides a Generic Security Services (GSS) library interface to Windows Security Support Provider Interface (SSPI). It uses a *.dll* to allow Kerberos security driver to use the Windows Security SSPI routines instead of the CyberSafe GSS libraries.

To use this feature, edit the *csfkrb5* entry in the *libtcl.cfg* file to include the *libsspiwrapper.dll*.

For example:

```
csfkrb5=LIBSKRB secbase=@REALM libgss=C:\sybase\OCS-12_5\lib3p\libsspiwrapper.dll
```

For more information on Kerberos security services, see *Open Client and Open Server Configuration Guide for Windows.*

Note Windows SSPI does not provide support for keytab files.

Credential delegation for MIT Kerberos

The Kerberos security driver now supports credential delegation when using the MIT Kerberos GSS library. This allows you to set up an Open Server gateway application that uses the delegated client credentials when establishing a connection with a remote server.

- **Establishing a connection with a remote server using credential delegation**

  This is an example of a call sequence you can employ when using credential delegation. You can refer to the ctos example in *$SYBASE/OCS-15_0/sample/srvlibrary*. *connect.c* now contains an example of the properties mentioned here:

  1 The client application requests for credential delegation and forwards the credential to the gateway connection using

    ```
    ct_con_props(..., CS_SET, SRV_SEC_DELEGATION, ...)
    ```

2 The connection handler of the gateway application checks whether the client requested credential delegation:

```
if (srv_thread_props(..., CS_GET,
     SRV_T_SEC_DELEGATION, ...))
    {...}
```

3 The connection handler retrieves the delegated client credentials:

```
srv_thread_props(..., CS_GET,
     SRV_T_SEC_DELEGCRED, ...)
```

4 The client application sets the delegated credentials in the Client-Library connection structure for use in connecting to the remote server:

```
ct_con_props(..., CS_SET, CS_SEC_CREDENTIALS, ...)
```

5 The client application attempts to connect to the remote server using ct_connect.

Requesting credential delegation using *isql* and *bcp*

You can also request for credential delegation using isql and bcp through the new isql and bcp sub-option for the -V parameter: -Vd. This new sub-option will request credential delegation and forward the client credentials to the gateway application. For example:

```
isql -Vd -SMY_GATEWAY
```

For detailed information on using credential delegation, see Open Server *Server-Library/C Reference Manual* and Open Client *Client-Library/C Reference Manual*.

New *isql* command line option *--retserverror*

You can use the new --retserverror option to force isql to terminate and return a failure code when it encounters a server error of severity greater than 10. In this type of abnormal termination, isql writes the label "Msg" together with the actual ASE error number to stderr, and returns a value of "2" to the calling program.

Syntax

```
isql --retserverror
```

Example

In this example, isql encounters a server error of severity 16. Since the --retserverror option is specified, isql returns "2" to the calling shell, prints "Msg 207" to stderr, and exits. As before, isql prints the full server error message to stdout. The same behavior applies to Windows, where you will find the return value of "2" in *%ERRORLEVEL%*:

```
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>
```

New *BCP* command line option *--skiprows*

You can use the new --skiprows option to specify that BCP must 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 is displayed.

Note --skiprows cannot co-exist with the -F option.

Syntax

```
bcp --skiprows nSkipRows
```

where, *nSkipRows* is the numbers of rows you want to skip.

Example

In this example, 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
```

ct_data_info() enhancement

You can now call ct_data_info() to retrieve fixed I/O fields such as object name *before* a a column is read. As before, the changeable fields in I/O descriptors such as pointers to text data, and length of text data are retrievable only after the column is read. This change is particularly useful when using srv_send_data() since srv_send_data() sends the row's data format before the whole row is read.

ADO.NET 2.0 support

ASE ADO.NET Data Provider 2.0 is shipped with this SDK release. The ADO.NET 2.0 features supported in this version are:

- Provider factories
- Provider statistics
- Bulk update
- Bulk copy
- Asynchronous commands
- Extended pooling support to clear pools
- Common base classes
- Database metadata

For more information on the supported ADO.NET 2.0 features, see What's New in ADO.NET 2.0 at http://msdn2.microsoft.com/en-us/library/ex6y04yf(VS.80).aspx.

BCP insert support

The jConnect Driver now supports large insertions of rows to ASE 12.5.2 and later versions using bulk load inserts. Although this feature does not require special configuration on the server, a larger page size, network packet size and max memory size significantly improves performance. Depending on the client memory, use of larger batches also improves performance.

ENABLE_BULK_LOAD

The enhanced jConnect Driver introduces a new property called ENABLE_BULK_LOAD which specifies whether or not to use bulk load to insert rows to the database. The values are:

- False – Disable bulk load. Default.
- True – Enable bulk load.

When you use prepared statements and ENABLE_BULK_LOAD is enabled, jConnect uses the BULK routines to insert a batch of records to the Sybase databases.

Limitation

These datatypes and features are not supported:

- unsigned types, bigint, and unitext
- partition tables, encrypted columns, and computed columns

Password expiration handling for ADO.NET, ODBC and OLE DB

Every company has a specific set of password policies for its database system. Depending on the policies, the password expires at a specific date and time. Unless the password is reset, the ASE drivers connected to a database throw password expired errors and suggest that the user change the password using isql. The password change feature enables users to change their expired passwords without having to use another tool.

Password expiration handling in ASE ADO.NET 2.0

ASE ADO.NET 2.0 supports the ChangePassword method which enables applications to change expired passwords without administrator intervention. For more information, see SqlConnection.ChangePassword Method at http://msdn2.microsoft.com/en-us/library/system.data.sqlclient.sqlconnection.changepassword(VS.80).aspx

Password expiration handling in ODBC and OLE DB

The ASE ODBC Driver and ASE OLE DB Provider introduce two connection string properties to support the password change feature:

- oldpassword – the current password. If oldpassword contains a value that is not null or an empty string, the current password is changed to the value contained in pwd.
- pwd – contains the value of the new password entered by the user. If oldpassword does not exist or is null, pwd contains the value of the current password.

ODBC Password change via connection dialog

A change password dialog is activated when "SQLDriverConnect with SQL_DRIVER_PROMPT" is set to True. In this dialog, the user is prompted for the current password and the new password that will replace it.

Mainframe Connect and DirectConnect for z/OS Option support

The ASE ODBC Driver by Sybase is now certified with the Mainframe Connect DirectConnect™ for z/OS Option. To support the DirectConnect for z/OS Option, two configuration properties—ServiceName and BackEndType—have been added to the driver.

ServiceName configuration property

The ServiceName property specifies the service name used to connect to the host. ServiceName can hold any string value. Its default value is an empty string ("").

BackEndType configuration property

The BackEndType configuration property specifies the target type of the DSN you are defining. The ODBC Driver can communicate with multiple targets including database systems like ASE and gateways to non-Sybase database systems. Currently, the ASE ODBC Driver supports these back end types:

- ASE (default)
- MFC Gatewayless
- DC DB2 Access Service
- DC TRS

For information on how the DirectConnect for z/OS Option uses the two new driver properties, see the Mainframe Connect DirectConnect for z/OS *Installation Guide*.

Updates and clarifications

This section lists updates made to previous ESDs.

Update to ESD #9

The section “Release of 32-bit binaries for 64-bit products on UNIX platforms” has been updated to include “Using strings to check the EBF number of 64-bit products” on page 65.

Update to ESD #8

ESD #8 has been updated to include the sections:

- “bcp discard file support for rejected rows” on page 66
- “Password expiration handling for jConnect for JDBC” on page 77

Modification to section “SSL Plus 5.0.4 + SBGSE 2.0”

Section “SSL Plus 5.0.4 + SBGSE 2.0” on page 143 has been modified to replace the dynamic load libray file *libsbgse2.dll* with *sbgse2.dll*.

New feature in ESD #9

This section describes the new feature in ESD #9.

Release of 32-bit binaries for 64-bit products on UNIX platforms

SDK/Open Server binaries like isql and bcp, share the same name between 32-bit and 64-bit products. Installing ASE, SDK, or Open Server 64-bit products with other Sybase 32-bit products overwrites the 32-bit binaries.

From 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.

Using *strings* to check the EBF number of 64-bit products

Since 32-bit binaries are released in 64-bit EBF, isql -v is no longer a valid command to check the EBF number for 64-bit products. Customers can use strings to confirm the EBF numbers for both Open Client and Open Server :

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

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

New features in ESD #8

This section describes features that are new for ESD #8.

Note ESQL/COBOL is now supported on HP Itanium 32-bit.

bcp discard file support for rejected rows

Originally, the *bcp* parameter *maxerrors* had to be set to a high value and the parameter *batchsize* set to 1, to identify the rows that were rejected due to exceptions such as duplicate rows or errors in the batch file. This method was not efficient and made it hard to identify, debug, and reload the rows that were rejected.

With the introduction of the new *bcp* option -d *discardfileprefix* you can now log the rejected rows into a dedicated discard file that has the same format as the host file. The discard file is created by appending the input file name to the discard file prefix supplied. You can correct the rows in this file and use it to reload the corrected rows.

Syntax

```
bcp -d discardfileprefix
```

Examples

This example creates the discard file *reject_titlesfile.txt*:

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

Usage

- 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.

New error messages

- "Unable to open the discard-file *discardfilename*."
- "I/O error while writing the bcp *discardfilename*."
- "Unable to close the file *discardfilename*. Data may not have been copied."

-e errorfile extended functionality

Currently, the bcp option -e *errorfile* logs the rows rejected due to conversion or format errors into an error file. ESD #8 extends the functionality of this option to log *all* rejected rows including those resulting from inserting duplicate rows when unique constraint is present, inserting into a table-partition that doesn’t fit the partition criteria, and truncation of data.

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

Note If -e *errorfile* is specified, the error messages appear on your terminal and are also logged in the error file.

Support for cached *interfaces* file

For Open Server and Open Client applications that use the *interfaces* file as a directory service, each time the applications build outgoing connections, a linked list is created in memory, and all entries of the interfaces file are loaded into this linked list. This causes an adverse effect on the performance due to excessive utilization of memory, if the number of outgoing connections is large and the *interfaces* file size is big.

The support for cached interfaces file addresses this issue. Caching the *interfaces* files in memory improves performance by eliminating the need to load the *interfaces* file from the hard disk for every outgoing connection. The *interfaces* file is loaded only when the first connection is built or when the *interfaces* file is updated.

Memory usage is also optimized because outgoing connections can share one linked list. When an *interfaces* file is updated, all connections built after the update will use the new *interfaces* file. The old *interfaces* file is kept in memory until all connections based on it are closed.

To use this new feature, rebuild your applications with the Open Server and Open Client ESD #8.

Support for ESQL structures and arrays as indicator variables

The current ESQL/C preprocessor supports the use of arrays as indicator variables but not the use of structures as indicator variables. In ESD #8, it is now possible to use ESQL/C structures as indicator variables. For more information, see “New array indicator feature in ESQL/C” on page 83.

The ESQL/COBOL preprocessor is also enhanced to support both array and structure indicator variables. For this feature to work correctly in COBOL, you must declare the indicator array or indicator structure elements with a PIC S9(4) clause and a COMP-5 clause. As with ESQL/C, use of structures and arrays as indicator variables removes the time consuming process of coding singleton indicator variables in ESQL/COBOL for every nullable column of every Embedded SQL statement in the application.

Examples

Following is an example of how to declare indicator structure in ESQL/C:

```
EXEC SQL BEGIN DECLARE SECTION;
/* Destination variables for fetches, using a struct.*/
 struct _hostvar {
        int m_titleid;
        char m_title[65];
```

```
        char m_pubname[41];
        char m_pubcity[21];
        char m_pubstate[3];
        char m_notes[201];
        float m_purchase;
 } host_var1;

/* An indicator structure for above variables. */
struct _indicvar {
       short i_titleid;
       short i_title;
       short i_pubname;
       short i_pubcity;
       short i_pubstate;
       short i_notes;
       short i_purchase;
 } indic_var1;
EXEC SQL END DECLARE SECTION;
```

Following is an example of how to declare indicator arrays, and how to execute a query using indicator arrays in ESQL/COBOL:

```
* Declare variables
....
01 HOST-STRUCTURE-M1.
   03 M-TITLE PIC X(64).
   03 M-NOTES PIC X(200).
   03 M-PUBNAME PIC X(40).
   03 M-PUBCITY PIC X(20).
   03 M-PUBSTATE PIC X(2).

01 INDICATOR-TABLE.
   03 I-NOTES-ARR PIC S9(4) COMP-5 OCCURS 5 TIMES.
....


* Execute query
....
EXEC SQL
SELECT substring(title, 1, 64), notes, pub_name,
            city, state
     INTO :HOST-STRUCTURE-M1:I-NOTES-ARR
     FROM titles, publishers
     WHERE titles.pub_id = publishers.pub_id
     AND title_id = :USER-TITLEID
END-EXEC.
....
```

Following is an example of how to declare indicator structures, and how to execute a query using indicator structures in ESQL/COBOL:

```
* Declare variables
....
01 HOST-STRUCTURE-M1.
   03 M-TITLE PIC X(64).
   03 M-NOTES PIC X(200).
   03 M-PUBNAME PIC X(40).
   03 M-PUBCITY PIC X(20).
   03 M-PUBSTATE PIC X(2).

01 INDICATOR-STRUCTURE-I1.
   03 I-TITLE PIC S9(4) COMP-5.
   03 I-NOTES PIC S9(4) COMP-5.
   03 I-PUBNAME PIC S9(4) COMP-5.
   03 I-PUBCITY PIC S9(4) COMP-5.
   03 I-PUBSTATE PIC S9(4) COMP-5.
....

* Execute query
....
EXEC SQL
SELECT substring(title, 1, 64), notes, pub_name, city,
       state
    INTO :HOST-STRUCTURE-M1:INDICATOR-STRUCTURE-I1
    FROM titles, publishers
    WHERE titles.pub_id = publishers.pub_id
    AND title_id = :USER-TITLEID
END-EXEC.
```

Usage

When using structs and arrays as indicator variables in ESQL/C and ESQL/COBOL:

- The number of elements in the indicator array or struct must be exactly the same as the number of elements in the host variable structure. A mismatch causes cpre or cobpre to stop processing, and code is not generated.
- The columns in the SELECT list must match by sequence, and datatype, the chosen structure name in the INTO list. A mismatch causes ct_bind() runtime errors and stops processing.

Error messages

Table 5 describes the new Embedded SQL internal error messages created to handle host variable versus indicator variable mismatch errors for this new feature.

Table 5: New internal error messages

Message ID	Message text	Severity	Fix
M_INVTYPE_V	Incorrect type of indicator variable found in the structure.	Fatal	Make sure that the same indicator variable is used in the hostvar and indicator declarations.
M_INVTYPE_VI	Mismatch between number of structure elements in the indicator structure and hostvar structure.	Fatal	Declare the same number of elements in the indicator structure and hostvar structure.
M_INVTYPE_VII	Mismatch between number of elements in the indicator array and hostvar structure.	Fatal	Declare the same number of elements in the indicator array and hostvar structure.

Limitations

You cannot mix singleton host variables or singleton indicator variables with hostvar structures, and indicator arrays or structures.

Open Client migration

Typically client applications connect to servers that provide services such as access to databases. Thus, when servers become unavailable, client applications disconnect. Although High Availability (HA) improves this behavior, in a true HA environment, seamless migration is necessary between servers. For example, when a server shuts down for scheduled maintenance, the Open Client migration feature allows Client-Library to migrate client connections to an available server. This feature is especially useful in a Shared Disk Cluster (SDC) environment. In addition to scheduled maintenance, the new feature can be used in cluster load balancing or when using special resources.

The Open Client migration feature works without the intervention of client applications, duplicating the client application's environment from the original server to the new server. The majority of client applications will not notice the change in servers.

To maximize the number of client applications that can be migrated, simply replace the Open Client shared libraries when using this feature.

Note DB-Library does not support connection migration.

Implementation

The Open Client migration feature is implemented as a new property with the default set to true. This occurs in all versions of Open Client. Upgrading Open Client to use the new feature depends on how the client application was built: If static libraries were used, the application must be relinked; if shared libraries were used, only shared libraries need be replaced.

You can disable the feature by setting the new CS_PROP_MIGRATABLE property to CS_FALSE using ct_config and ct_con_props.

Note This feature is only available when connected to a server that supports connection migration.

Extended password encryption

Note To use the extended password encryption feature, you require a server that supports extended password encryption, such as ASE 15.0.2.

The extended password encryption feature enhances the existing symmetrical key password encryption provided by the drivers and data providers to support asymmetrical key encryption. It is supported for the following drivers and providers:

- ADO.NET Data Provider
- ODBC Driver
- OLE DB Data Provider
- jConnect for JDBC Driver

The symmetrical key encryption mechanism uses the same key to encrypt and decrypt the password and is therefore considered a weak encryption mechanism. An asymmetrical key encryption mechanism uses one key (called the public key) to encrypt the password and another key (called the private key) to decrypt the password. Because the private key is not shared across the network, the assymetrical key encryption is considered more secure.

Note When using the extended password encryption feature, you may experience a slight delay in login time due to the additional processing time required for asymmetrical encryption.

Enabling extended password encryption for ASE ODBC, OLE DB, and ADO.NET

This section describes how to enable extended password encryption for the ASE ODBC, OLE DB, and ADO.NET Drivers and Providers.

Using the EncryptPassword Connection string property

The EncryptPassword connection property specifies whether the password is transmitted in encrypted format. In ESD #8, this same property is used to enable asymmetric key encryption, if available. When password encryption is enabled, and the server supports asymmetric key encryption, this format is used instead of the symmetric key encryption. The new EncryptPassword values are:

- 0 – Use plain text password (the default value).
- 1 – Use encrypted password. If it is not supported, return an error message.
- 2 – Use encrypted password. If it is not supported, use plain text password.

Note If the server is configured to require clients to use an encrypted password, entering a plain text password will cause login to fail.

Using *odbc.ini* file on Linux

Following is an example of how you can specify the use of an encrypted password in the *odbc.ini* file:

```
[sampledsn]
Description=Sybase ODBC Data Source
UserID=abc
Password=xyz
```

```
Driver=Adaptive Server Enterprise
Server=sampleserver
Port=4100
Database=pubs2
UseCursor=1
EncryptPassword=1
```

Using the ODBC Data Source Administrator on Windows

To enable password encryption for a data source in the ODBC Data Source Administrator, check the Encrypt Password checkbox in the Connection tab page when setting up the data source.

Note You can only enable or disable password encryption (which corresponds to EncryptPassword value of 1 and 0, respectively) from the user interface. You can set EncryptPassword to 2 from a connection string.

Using the Sybase ASE Data Source Administrator for OLE DB Connections

To enable password encryption for an OLE DB data source in the Sybase ASE Data Source Administrator, check the Encrypt Password checkbox in the Connection tab page when setting up the data source.

Note You can only enable or disable password encryption (which corresponds to EncryptPassword value of 1 and 0, respectively) from the user interface. You can set EncryptPassword to 2 from a connection string.

Enabling extended password encryption for the jConnect for JDBC Driver

This section describes how to enable extended password encryption for the jConnect for JDBC Driver.

Using the ENCRYPT_PASSWORD connection property,

The ENCRYPT_PASSWORD connection property specifies whether the password is transmitted in encrypted format. In ESD #8, this same property is used to enable asymmetric key encryption. When password encryption is enabled and the server supports asymmetric key encryption, this format is used instead of the symmetric key encryption.

Set the ENCRYPT_PASSWORD connection property to true to enable password encryption. The default value is false.

Note If the server is configured to require clients to use an encrypted password, entering a plain text password will cause login to fail.

Enabling login retry with a clear text password

Server login fails when the ENCRYPT_PASSWORD property is set to true, and the server does not support password encryption. If you want to use a clear text password for servers that do not support password encryption, set the RETRY_WITH_NO_ENCRYPTION connection property to true.

When both ENCRYPT_PASSWORD and RETRY_WITH_NO_ENCRYPTION properties are set to true, jConnect first logs in using the encrypted password. If login fails, jConnect logs in using the clear text password.

Setting up the Java Cryptography Extension (JCE) provider

The new asymmetric password encryption mechanism uses RSA encryption algorithms to encrypt the password being transmitted. In order to perform this RSA encryption, configure your JRE with a suitable Java Cryptography Extension (JCE) provider. The configured JCE provider should be capable of supporting the “RSA/NONE/OAEPWithSHA1AndMGF1Padding” transformation.

The Sun JCE provider included with Sun JREs may not be capable of handling the “RSA/NONE/OAEPWithSHA1AndMGF1Padding” transformation. In order to use the extended password encryption feature in this case, you need to configure an external JCE provider that includes support for this transformation. If the JCE is not capable of handling the required transformation, you will receive an error message at login.

You can use the JCE_PROVIDER_CLASS connection property to specify the JCE provider. There are a number of commercial and open source JCE providers that you can choose from. For example, the “Bouncy Castle Crypto APIs for Java” is a popular open source Java JCE provider. If you choose not to specify the JCE_PROVIDER_CLASS property, jConnect will attempt to use any bundled JCE.

To specify a JCE provider:

- Set the JCE_PROVIDER_CLASS property to the fully qualified class name of the provider you want to use. For example, to use the Bouncy Castle JCE:

```
String url = "jdbc:sybase:Tds:myserver:3697";
Properties props = new Properties();
props.put("ENCRYPT_PASSWORD ", “true”);
props.put("JCE_PROVIDER_CLASS",
"org.bouncycastle.jce.provider.BouncyCastleProvider
");

/* Set up additional connnection properties as
needed */
props.put("user", "xyz");
props.put("password", "123");

/* get  the connection */
Connection con = DriverManager.getConnection(url,
props);
```

- Configure the JCE provider before using it. This can be done by one of two ways:
  - Copy the JCE provider *jar* file into the JRE standard extension directory:
    - For UNIX / Mac OS X platforms:
      ${JAVA_HOME}/jre/lib/ext
    - For Windows:
      %JAVA_HOME%\jre\lib\ext
  - If you cannot copy the JCE *jar* file to the appropriate directory, refer to the Sun JCE Reference Guide at http://java.sun.com/j2se/1.4.2/docs/guide/security/jce/JCERefGuide.html for instructions on setting up an external JCE provider.

If jConnect is unable to use the JCE provider specified, it will attempt to use the JCE providers configured in the JRE security profile. If no other JCE providers are configured or configured providers do not support the required transformation and password encryption is enabled, the connection will fail.

Note The JCE provider you specify must support the “RSA/NONE/OAEPWithSHA1AndMGF1Padding” transformation.

Password expiration handling for jConnect for JDBC

Originally, whenever the database password expires, jConnect for JDBC would throw an exception to the client, and would suggest to change the password using isql. With the password expiration handling feature, jConnect handles the expired password exceptions and changes the password internally. jConnect for JDBC introduces two new connection properties to support this feature—PROMPT_FOR_NEWPASSWORD and NEWPASSWORD.

When a client password expires, jConnect checks the value of PROMPT_FOR_NEWPASSWORD to determine whether to do a transparent password change or to prompt for the new password. If this property is set to True, a dialog box pops up and the user is prompted to set the new password manually. If PROMPT_FOR_NEWPASSWORD is set to False, jConnect checks the value of NEWPASSWORD and, if NEWPASSWORD is not null, uses this value to replace the expired password.

The default value of PROMPT_FOR_NEWPASSWORD is false while the default value of NEWPASSWORD is null.

Extended support for secure LDAP connections using SSL/TLS

It is now possible to set up a secure connection to an LDAP Directory Server using SSL or TLS on both 32-bit and 64-bit platforms. For more information, see the Open Client Client-Library/C *Reference Manual.*

Extended platform support for ASE OLE DB Provider

A 64-bit version of the ASE OLE DB Provider by Sybase is now supported on Windows x64 platforms.

New features in ESD #7

This section describes features that are new for ESD #7.

Extended password encryption

The extended password encryption feature enhances the existing symmetrical key type password encryption provided with Open Client libraries.

A symmetrical key type uses the same key to encrypt and decrypt the password. An asymmetrical key type uses one key to encrypt the password and another to decrypt the password. The new enhanced password encryption uses an asymmetrical key type instead of the original symmetrical key type. This allows Sybase applications to transmit strong public key passwords via networks providing a secure handshake protocol for password-based authentication.

Note To use the extended password encryption feature, you require a server with strong password encryption support, such as ASE 15.0.2.

Note When using the extended password encryption feature, you may experience a slight 1-2 second delay in login time. This is because of the additional processing time required for asymmetrical encryption.

CS_SEC_EXTENDED_ENCRYPTION

A new connection property, CS_SEC_EXTENDED_ENCRYPTION, is provided to enable or disable the new extended password encryption feature. CS_SEC_EXTENDED_ENCRYPTION is set to CS_FALSE by default. You must use ct_con_props to enable the feature by setting the value to CS_TRUE.

CS_SEC_ENCRYPTION is the connection property for normal password encryption. CS_SEC_ENCRYPTION is also set to CS_FALSE by default.

If an Open Client application logs onto a server with both CS_SEC_EXTENDED_ENCYPTION and CS_SEC_ENCRYPTION set to CS_TRUE, it uses extended password encryption as the first preference.

If your server cannot support extended password encryption, it uses normal password encryption. If your server cannot support both extended and normal encryption, it fails the connection request and reconnects using a plaintext password.

Syntax

```
CS_RETCODE ct_con_props (CS_CONNECTION *connection,
                         CS_INT action, CS_INT property,
                         CS_VOID *buffer, CS_INT buflen,
                         CS_INT* outlen)
```

Parameters

connection

Pointer to CS_CONNECTION structure.

action

Symbolic values for CS_SET, CS_GET, CS_CLEAR, or CS_SUPPORTED.

property

Symbolic name for CS_SEC_EXTENDED_ENCRYPTION.

buffer

Pointer to CS_TRUE or CS_FALSE values.

buflen

Fixed length values passed as CS_UNUSED.

outlen

Unused value passed as NULL.

Example

In the following example, CS_SEC_EXTENDED_ENCRYPTION is disabled:

```
...
CS_INT Ex_encryption = CS_FALSE;
CS_INT Ex_nonencryptionretry = CS_FALSE;

...
main()
{
   ...

   /*
   ** This needs to be called before calling ct_connect()
   */
   ret = ct_con_props(connection, CS_SET, CS_SEC_EXTENDED_ENCRYPTION,
          &Ex_encryption, CS_UNUSED, NULL);
   EXIT_ON_FAIL(context, ret, "Could not set extended encryption");

   ret = ct_con_props(connection, CS_SET, CS_SEC_NON_ENCRYPTION_RETRY,
          &Ex_nonencryptionretry, CS_UNUSED, NULL);
   EXIT_ON_FAIL(context, ret, "Could not set non encryption retry");

   ...
}
```

CS_SEC_NON_ENCRYPTION_RETRY

A new connection property, CS_SEC_NON_ENCRYPTION_RETRY, is provided to enable or disable plain text password retries when your server cannot support extended password encryption or normal password encryption.

By default, CS_SEC_NON_ENCRYPTION_RETRY is set to CS_TRUE and you use ct_con_props to enable or disable the feature. If extended password encryption or normal password encryption is enabled, ct_con_props automatically disables CS_SEC_NON_ENCRYPTION_RETRY.

The syntax, parameters and example for CS_SEC_NON_ENCRYPTION_RETRY are the same as that provided for "CS_SEC_EXTENDED_ENCRYPTION" on page 78.

CS_EXTENDED_ENCRYPT_CB

A callback handler type, CS_EXTENDED_ENCRYPT_CB, is provided to set the extended password encryption feature.

CS_EXTENDED_ENCRYPT_CB allows you to set the feature using the non-default public key encryption handler. The default handle performs the encryption expected by ASE as most applications connecting directly to ASE or to ASE via an Open Server gateway rely on default encryption.

Alternatively, CS_EXTENDED_ENCRYPT_CB sets the feature with the application's public key encryption handler installed with ct_callback.

Syntax

```
CS_RETCODE ct_callback(CS_CONTEXT *context,
                       CS_CONNECTION *connection,
                       CS_INT action, CS_INT type,
                       CS_VOID *func)
```

Parameters

context

Pointer to CS_CONTEXT structure.

connection

Pointer to CS_CONNECTION structure.

Note Either *context* or *connection* must be NULL.

---

action

Symbolic value for CS_SET or CS_GET.

type

Symbolic name for CS_EXTENDED_ENCRYPT_CB.

func

Function pointer to encryption callback function.

SRV_NEG_EXTENDED_ENCRYPT, SRV_NEG_EXTENDED_LOCPWD, and SRV_NEG_EXTENDED_REMPWD

Three Open Server library negotiation types, SRV_NEG_EXTENDED_ENCRYPT, SRV_NEG_EXTENDED_LOCPWD, and SRV_NEG_EXTENDED_REMPWD, are provided for Open Server applications to send and receive data from a client.

Calling srv_negotiate(CS_SET, SRV_NEG_EXTENDED_ENCRYPT) is used to send to a client the negotiated login information and public key used to encrypt the password.

Calling srv_negotiate (CS_GET, SRV_NEG_EXTENDED_LOCPWD) allows an Open Server application to get the public key encrypted password that was sent by the client.

Calling srv_negotiate (CS_GET, SRV_NEG_EXTENDED_REMPWD) allows an Open Server application to get the variable number of pairs of remote server names and corresponding public key encrypted passwords sent by the client.

The usage of srv_negotiate sending SRV_NEG_EXTENDED_ENCRYPT and receiving SRV_NEG_EXTENDED_LOCPWD/SRV_NEG_EXTENDED_REMPWD is as follows:

Syntax

CS_RETCODE **srv_negotiate** (SRV_PROC **spp*, CS_INT *cmd*, CS_INT *type*)

Parameters

spp

Pointer to internal thread control structure.

cmd

Symbolic value for CS_SET or CS_GET.

type

Symbolic name for SRV_NEG_EXTENDED_ENCRYPT (CS_SET only) or SRV_NEG_EXTENDED_LOCPWD/SRV_NEG_EXTENDED_REMPWD (CS_GET only).

SRV_EXTENDED_ENCRYPT

A new Open Server library bit mask SRV_EXTENDED_ENCRYPT is provided, allowing server applications to obtain negotiated properties of server threads used with the extended password encryption feature. srv_thread_props(CS_GET, SRV_T_NEGLOGIN) is used to obtain negotiation properties of server threads.

Syntax

CS_RETCODE **srv_thread_props**(SRV_PROC *spp, CS_INT *cmd*,
CS_INT *property*, CS_VOID *bufp,
CS_INT *buflen*, CS_INT *outlenp*)

Parameters

spp

Pointer to internal thread control structure.

action

Symbolic value for CS_GET.

property

Symbolic name for SRV_T_NEGLOGIN.

buffer

Pointer to CS_INT variable. This CS_INT is the bit mask "or" value of all negotiation types. All possible values for SRV_T_NEGLOGIN are:

SRV_CHALLENGE ((CS_INT)0X0001)

SRV_ENCRYPT ((CS_INT)0X0002)

SRV_SECLABEL ((CS_INT)0X0004)

SRV_APPDEFINED ((CS_INT)0X0008)

SRV_EXTENDED_ENCRYPT ((CS_INT)0X0010)

buflen

Size of CS_INT.

outlen

Unused parameter that should be passed as NULL.

Sybase customized Open SSL support

SSL functionality is supported using Open SSL for the following platforms:

- Windows 2003 (x64)

- Linux on POWER (32-bit and 64-bit)

To enable the SSL functionality, add the following runtime libraries to *libtcl.cfg* (32-bit) or *libtcl64.cfg* (64-bit) configuration files:

For Linux on POWER:

- *libsybfcsissl.so.15.0.3* (32-bit)
- *libsybfcsissl64.so.15.0.3* (64-bit)

For Windows:

- *libsybfcsissl64.dll*

Configuration files are available in the following directories:

- *%SYBASE%\%SYBASE_OCS%\ini* (Windows)
- *$SYBASE/$SYBASE_OCS/config* (Linux on POWER)

Note For more information on SSL, refer to the Open Client *Client-Library/C Reference Manual*.

New array indicator feature in ESQL/C

With current ESQL/C preprocessors, it is not possible to use an array as an indicator variable. It only allows you to use indicator variables that tie to (nullable) host variables. Furthermore, each indicator variable must be coded for each nullable column in every embedded SQL statement of that application.

The new array indicator feature in ESQL/C removes these restrictions, allowing you to use an array of shorts for the indicators combined with a structure. This holds the host variables that can be subsequently referenced in a SQL statement.

Examples

The following example describes how to declare the new array indicator feature:

```
EXEC SQL BEGIN DECLARE SECTION;
/* Destination variables for fetches, using a struct. */

struct _vararr {
    int  m_titleid;
    char m_title[65];
    char m_pubname[41];
    char m_pubcity[21];
```

```
    char m_pubstate[3];
    char m_notes[201];
    float m_purchase;
} var_array;

/* An indicator array for all variables. */

short i_notes[7];

EXEC SQL END DECLARE SECTION;
```

The following example describes how to execute a query of the new array indicator:

```
EXEC SQL
SELECT titleid, title, pubname, city, state, notes,
        purchases
INTO  :var_array INDICATOR :i_notes
FROM  T1, T2
WHERE .....
```

Where:

- `INTO`

  `:var_array` is the reference to the hostvar structure that holds the program's column variables at fetch time.

- `INDICATOR`

  `INDICATOR` is an optional keyword, and can be omitted. However, `:i_notes` in the example is the reference to the indicator array, and the `:` is mandatory.

  The indicator array is referenced by name, and no individual elements must be used. The number of elements in the indicator array must also match the number of host variables in the `var_array` structure.

- `SELECT`

SELECT is the (column) select list from the selected tables. The select list must match by position and datatype the host variables in the INTO list. Failure to match the SELECT and INTO list causes runtime errors in the application.

Note Though the example is based on a two-table join, SELECT and INTO must still match.

Alternatively, for single-table queries, entering * expands to the same number of columns and datatypes as is depicted in INTO.

Note SELECT * can have an adverse effect on (network) performance, particularly if the select list expands to a large number of columns.

Extended BCP support for encrypted columns

Originally, ASE encrypted columns relied on the ASE permission system to protect encrypted data. This required decrypt permissions to reference encrypted columns in selected target lists and did not prevent unauthorized administrators from accessing data.

The new feature allows keys and encrypted columns to be protected with passwords supplied by non-administrators. The passwords are used by ASE 15.0.2 to access encrypted data and are set by entering:

```
set encryption passwd <password> for [key | column]
<key_name | column_name>
```

The new feature supports encrypted columns in bcp, which is the bulk copy utility used to move data between ASE and the Operating System (OS) file.

Originally, bcp with -C allowed copying of encrypted column data in the encrypted format. Correspondingly, bcp without -C allowed copying of encrypted column data in the decrypted format.

The new feature requires passwords for bcp to copy decrypted formats to OS files. Conversely, unencrypted data from an OS file can be copied into an encrypted column in encrypted format.

Note To use the new feature, you require a database server with extended support for encrypted columns, such as ASE 15.0.2.

bcp syntax changes

As illustrated below, two arguments, --colpasswd column_name password and --keypasswd key_name password, are added for the bcp utility to obtain decrypt permissions for encrypted columns:

```
bcp ...
    [--colpasswd [[[db_name.[owner].]table_name.]column_name
[password]]]
    [--keypasswd [[db_name.[owner].]key_name [password]]]
...
    [-m maxerrors] [-f formatfile] [-e errfile]
    [-F firstrow] [-L lastrow] [-b batchsize]
    [-n] [-c] [-t field_terminator] [-r row_terminator]
    [-U username] [-P password] [-I interfaces_file] [-S server]
    [-a display_charset] [-z language] [-v]
    [-A packet size] [-J client character set]
    [-T text or image size] [-E] [-g id_start_value] [-N] [-X]
    [-M LabelName LabelValue] [-labeled]
    [-K keytab_file] [-R remote_server_principal] [-C]
    [-V [security_options]] [-Z security_mechanism] [-Q] [-Y]
    [-x trusted.txt_file]
    [--maxconn maximum_connections] [--show-fi] [--hide-vcc]
    [--colpasswd [[[db_name.[owner].]table_name.]
       column_name [password]]]
    [--keypasswd [[db_name.[owner].]key_name [password]]]
    [--initstring ASE initialization string]
```

The new arguments enable the bcp utility to bulk copy encrypted columns data (ciphertext) to and from OS files in the decrypted format (plaintext).

--colpasswd column_name password allows you to set 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. You must supply the password a second time to access the second column.

--keypasswd key_name password allows you to set passwords for all columns accessed by a key by sending set encryption passwd <*password*> for key <*key_name*> to ASE.

Examples

Example 1 To set the password to `pwd1` for encrypted column `col1`:

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

Example 2 To set a prompt to enter the password for encrypted column `col1`:

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

Example 3 To read 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 4 To set the password `pwd1` for encrypted column `col1` and password `pwd2` for encrypted column `col2`:

```
bcp mydb..mytable out myfile –U uuu –P ppp
    --colpasswd db..tbl.col1 --colpasswd db..tbl.col2
Enter column db..tbl.col1's password: ***?
Enter column db..tbl.col2's password: ***?
```

Example 5 To set password `pwd1` for encryption key `key1`:

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

Example 6 To set a prompt to enter the password for encryption key `key1`:

```
bcp mydb..mytable in myfile –U uuu –p ppp
    --keypasswd db..key1
Enter key db..key1's password: ***?
```

Example 7 To read the password for encryption key `KEY1` from external OS file `PASSWORDFILE`:

```
BCP MYDB..MYTABLE IN MYFILE –U UUU –P PPP
    --KEYPASSWD DB..KEY1 < PASSWORDFILE
```

Example 8 To set password `pwd1` for encryption key `key1` and password `pwd2` for encryption key `key2`:

```
bcp mydb..mytable in myfile –U uuu –p ppp
    --keypasswd db..key1 --keypasswd db..key2
Enter key db..key1's password: ***?
Enter key db..key2's password: ***?
```

Support for secure LDAP connections using SSL/TLS

It is now possible to set up a secure connection to an LDAP Directory Server using SSL or Transport Layer Security (TLS). To establish a secure connection between a client and a LDAP Directory Server, use either of the following methods:

- Establish a secure connection to the secure port of the LDAP Server. This is typically port number 636 and is established by entering the following syntax in the *libtcl.cfg* file:

  ```
  [DIRECTORY]
  ldap=libsybdldap.so
  ldaps://huey:636/dc=sybase,dc=com????bindname=cn=Manager,dc=Sybase,dc=com?secret
  ```

  If no port number is specified with ldaps://, port number 636 is used by default.

- Upgrade a normal connection (typically port number 389 of the LDAP Server) to a secure one, using StartTLS. To upgrade the connection, enter the following syntax in the *libtcl.cfg* file:

  ```
  [DIRECTORY]
  ldap=libsybdldap.so starttls
  ldap://huey:11389/dc=sybase,dc=com????bindname=cn=Manager,dc=Sybase,dc=com?secret
  ```

  If no port number is specified with ldap://, port number 389 is used by default.

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

Note LDAP connections using SSL/TLS are currently only available for 32-bit platforms.

Extended support for OpenLDAP

OpenLDAP is now supported on Windows platforms.

New *srv_send_data* routine added

Description

The new API routine, srv_send_data, allows Open Server applications to transfer rows containing multiple columns to clients. It allows Open Server applications to send text and image data in chunks, preventing excessive utilization of memory.

Syntax

```
CS_RETCODE srv_send_data(spp, column, buf, buflen)

SRV_PROC  *spp;
CS_INT        *column;
CS_BYTE      *buf;
CS_INT         buflen;
```

Parameters

spp

A pointer to an internal thread control structure.

column

The number of the column in a row set.

buf

A pointer to a buffer containing the data to send to the client. This determines the size of a section.

buflen

The length of the **buf* buffer.

Return Value

Table 6: Return values (srv_send_data)

Returns	To indicate
CS_SUCCEED	The routine completed successfully.
CS_FAIL	The routine failed.

Example

```
#include <ctpublic.h>
#include <ospublic.h>
/*
** Local Prototype.
*/
CS_RETCODE ex_srv_send_data PROTOTYPE((
    SRV_PROC *spp,
    CS_COMMAND *cmd,
    CS_INT cols
));
#define MAX_BULK 51200

/*
** EX_SRV_SEND_DATA
```

```
** Example routine to demonstrate how to write columns
** of data in a row set to a client using srv_send_data.
** This routine will send all the columns of data read
** from a server back to the client.

** Arguments:
** spp A pointer to an internal thread control
** structure.
** cmd The command handle for the command that is
** returningtext data.
** cols The number of columns in a row set.

** Returns:
** CS_SUCCEED Result set sent successfully to client.
** CS_FAIL An error was detected.
*/
CS_RETCODE ex_srv_send_data(spp, cmd, cols)
SRV_PROC *spp;
CS_COMMAND *cmd;
CS_INT cols;
{
    CS_INT *len;/* Length of column data. */
    CS_INT *outlen;/* Number of bytes received. */
    CS_BYTE **data;/* Column data. */
    CS_BYTE buf[MAX_BULK]; /* Buffer for text data. */
    CS_BOOL ok; /* Error control flag. */
    CS_INT  i;
    CS_INT  ret;

    /* Initialization. */
    ok = CS_TRUE;

    /*
    ** Transfer a row.
    */
    for (i = 0; i < cols; i++)
    {
       if ((fmt[i].datatype != CS_TEXT_TYPE) &&
           (fmt[i].datatype != CS_IMAGE_TYPE))
       {
           /*
           ** Transfer a non TEXT/IMAGE column.
           */

           /*
           ** Read the data of a non-text/image column
```

```
            ** from the server.
            */
            ret = ct_get_data(cmd, i+1, data[i],
                len[i], &outlen[i]);
            if ((ret != CS_SUCCEED)
                && (ret != CS_END_DATA)
                && (ret != CS_END_ITEM))
            {
                ok = CS_FALSE;
                break;
            }

            /*
            ** Write the data of a non-text/image column
            ** to client.
            */
            if (ret = srv_send_data(srvproc, i+1,
                    NULL, 0) != CS_SUCCEED)
            {
                ok = CS_FALSE;
                break;
            }
        }
        else
        {
            /*
            ** Transfer a TEXT/IMAGE column in small
            ** trunks.
            */

            /*
            ** Read a chunk of data of a text/image column
            ** from the server.
            */
            while ((ret = ct_get_data(cmd, i+1, buf,
                MAX_BULK, &len[i])) == CS_SUCCEED)
            {
                /* Write the chunk of data to client. */
                if (ret = srv_send_data(srvproc, i+1, buf,
                    len[i]) != CS_SUCCEED)
                {
                    ok = CS_FALSE;
                    break;
                }
            }
        }
```

```
    }

    switch(ret)
    {
    case CS_SUCCEED:
       /* The routine completed successfully. */
    case CS_END_ITEM:
       /* Reached the end of this item's value. */
    case CS_END_DATA:
       /* Reached the end of this row's data. */
    break;
    case CS_FAIL:
       /* The routine failed. */
    case CS_CANCELED:
       /* The get data operation was cancelled. */
    case CS_PENDING:
       /* Asynchronous network I/O is in effect. */
    case CS_BUSY:
       /* An asynchronous operation is pending. */
    default:
    ok = CS_FALSE;
    }
    return (ok ? CS_SUCCEED : CS_FAIL);
}
```

Note For more information on using srv_send_data, refer to ctos_procmultitextcol in the *ctos.c* sample program.

Usage

- srv_send_data is used to send data of a row set column by column to a client.
- When sending columns with text or image data, Open Server applications must call srv_text_info before srv_send_data. This ensures the data stream is correctly set to the total length of data being sent. The application then calls srv_send_data to send the data in chunks, and continues to call the routine until there is no remaining data to be sent.
- Open Server applications can send text and image data to clients using srv_bind and srv_xferdata. However, these routines require all data columns to be sent at once. srv_send_data allows applications to send text and image data in chunks.
- Open Server applications treat text and image data streams identically, with the exception of character set conversions. These conversions are only performed on text data.

See also

Related srv_bind, srv_get_text, srv_text_info, srv_xferdata, srv_get_data, and srv_send_text routines in the Open Server 15.0 *Server Library/C Reference Manual*.

New features in ESD #6

This section describes features that are new for ESD #6.

New LDAP Directory Server retry and delay options

Connections or directory lookups to the LDAP Directory Server may be unable to complete due to a hanging or otherwise unavailable LDAP Server.

The Timeout support for LDAP feature in ESD #5 introduced time limits on failed connections or lookups to the LDAP Directory Server.

In ESD #6, this feature is enhanced with retry and delay options that allow you to specify the number of times to retry connections to the LDAP Directory Server, and delays between the retries.

New retry and delay options

The retry option specifies the number of times to retry a search connection to the LDAP Directory Server after the initial attempt fails or times out. The delay option is the number of seconds to wait between a failed and new retry. Both options are set in *libtcl.cfg* and apply to the designated LDAP Directory Server only, for example:

```
[DIRECTORY]
myldap=libsybdldap.so retry=3 delay=5
ldap://nlnognix/dc=sybase,dc=com????bind…
```

By default, both options are 0.

Unsafe Null with indicator variables in ESQL/C

By default, current ESQL/C and ESQL/COBOL preprocessors (*cpre* and *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 now avoid these issues by using a new command line flag, -u, with *cpre* and *cobpre*. -u generates code that disables ANSI binds. You may also disable ANSI binds by setting CS_ANSI_BINDS to CS_FALSE in the *ocs.cfg* file.

New logging feature for ASE OLE DB Provider

The new logging feature for ASE OLE DB Provider allows you to track an application's database access by logging calls to the OLE DB public API.

You enable the logging feature by creating a registry entry that identifies a configuration file. The configuration file is a properties file that allows you to log all calls to OLE DB, or limit logging to a specified subset of the API.

❖ **To enable the logging feature**

1 Create a properties file. For example:

```
### --- Begin oledblog.properties ---
#---------------------------------
#Configure loggers

# by default turn everything off. Change this to "=TRACE, OLEDB_LOGFILE"
# to default to logging to a file
log4cplus.rootLogger=OFF, NULL

# We want to log the OLEDB API calls to a log file
# change this to ="OFF,NULL" if you only want to log certain methods
log4cplus.logger.com.sybase.dataaccess.oledb=TRACE, OLEDB_LOGFILE
log4cplus.additivity.com.sybase.dataaccess.oledb=false

# We don't want to log the IUnknown::AddRef and Release calls
log4cplus.logger.com.sybase.dataaccess.oledb.IUnknown.AddRef=OFF, NULL
log4cplus.additivity.com.sybase.dataaccess.oledb.IUnknown.AddRef=false
log4cplus.logger.com.sybase.dataaccess.oledb.IUnknown.Release=OFF, NULL
log4cplus.additivity.com.sybase.dataaccess.oledb.IUnknown.Release=false

# Other OLEDB interface can be turned on and off as well
```

```
# You can turn on (or off) logging of all methods on a given interface
# by configuring:
# log4cplus.logger.com.sybase.dataaccess.oledb.<InterfaceName>=???

# You can get more specific and only log (or turn off logging for) a
# specific method:
log4cplus.logger.com.sybase.dataaccess.oledb.<InterfaceName>.
<MethodName>=TRACE, OLEDB_LOGFILE


#-------------------------------------
#Configure logging appenders

# Throw away all log messages
log4cplus.appender.NULL=log4cplus::NullAppender

# Send log messages to the console
log4cplus.appender.STDOUT=log4cplus::ConsoleAppender
log4cplus.appender.STDOUT.layout=log4cplus::PatternLayout
### Modify this line to define the output format
log4cplus.appender.STDOUT.layout.ConversionPattern=%m%n

# Write the log messages to a file
log4cplus.appender.OLEDB_LOGFILE=log4cplus::FileAppender
log4cplus.appender.OLEDB_LOGFILE.layout=log4cplus::PatternLayout
log4cplus.appender.OLEDB_LOGFILE.ImmediateFlush=true
### Modify this line to define the output format
log4cplus.appender.OLEDB_LOGFILE.layout.ConversionPattern=%m%n
### Modify this line to specify the file to save the log to.
log4cplus.appender.OLEDB_LOGFILE.File=c:\temp\oledb.log


### --- End oledblog.properties ---
```

Note For more information on creating a properties file, refer to the log4cplus project page at http://log4cplus.sourceforge.net.

2 Create a registry entry:

HKEY_CURRENT_USER\Software\Sybase\OLEDB Provider

LogConfigFile=<*path to properties file*>

3 Restart the application you want to log. Upon restart, the application begins to log to the log file specified in the properties file.

Note To disable the logging feature, delete the registry entry created in Step 2.

Updates and clarifications

Adaptive Server Enterprise (ASE) 15.0 optimizer options introduced in ESD #5 are no longer available. These options were described in the "New optimizer options" section, in the ESD #5 Open Server 15.0 and SDK 15.0 *New Features* for Microsoft Windows, Linux, and UNIX (DC20155-01-1500-09). Please disregard this section, if referring to the ESD #5 *New Features*.

New features in ESD #5

This section describes features that are new for ESD #5.

Timeout support for LDAP

You can now specify a time limit for LDAP Directory Server lookups. Exceeding the time limit results in LDAP Directory Server lookups timing out.

The value of the time limit is set with the CS_DS_TIMELIMIT connection property. Refer to the Open Client *Client-Library/C Reference Manual* for instructions on how to set the CS_DS_TIMELIMIT connection property using ct_con_props.

Note If you do not set CS_DS_TIMELIMIT, the connection property uses the login timeout value as the default time limit for LDAP Directory Server lookups.

Large identifier capability check in Open Server

Open Server now checks to see if connected clients have the capability to support large identifiers. If clients do not support large identifiers, and request objects with names larger than 132 bytes (the pre-15.0 identifier limit for Client-Library), Open Server displays the following errror:

```
Identifier starting with <name> is too long. Client does
not support Large Identifiers.
```

where *name* is the name of the identifier.

BCP support for initialization strings

You can now use the bcp command to send Transact-SQL commands, such as set replication off, to Adaptive Server before data is transferred.

Although you can use any Transact-SQL command 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.

bcp syntax changes

The following BCP parameter is included to support Transact-SQL initialization commands:

-- initstring "*Transact-SQL command*"

Note Result sets issued by the initialization string are silently ignored, unless any error occurs.

Example

In the following example, replication is disabled 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.

Note If Adaptive Server returns an error, BCP stops the data transfer and displays an error message.

RPC support for large identifiers

Although Open Client and Open Server have no explicit limits on the length of an RPC name, the length of the name field (NameLen) in the TDS_DBRPC token is 1 byte. This implicitly limits the length of a Remote Procedure Call (RPC) name to 255 bytes.

To overcome this limitation, Open Client and Open Server now support the TDS token, TDS_DBRPC2.

TDS_DBRPC2 is an enhanced version of TDS_DBRPC and provides support for RPC name lengths that are up to 2 bytes (unsigned) in length.

A new capability bit, CS_REQ_DBRPC2, and its TDS equivalent, TDS_REQ_DBPRC2, are also provided.

The following sections describe changes to Open Client and Open Server in supporting large identifiers for RPC.

Open Client updates

ct_command check

A check is run at ct_command, determining if TDS_DBRPC2 is enabled or disabled.

- If TDS_DBRPC2 is enabled, the maximum length of an RPC name is 65531 bytes.
- If TDS_DBRPC2 is disabled, the maximum length is set at 255 bytes.

New capability tests

If RPC name length is less than 256 bytes, TDS_DBRPC is always used. For RPC's that exceed 255 bytes, the TDS_DBRPC2 token is used, but only if CS_REQ_DBRPC2 capability is on.

Open Server updates

Limits on RPC name components

Although the maximum length of an RPC name can now be greater than 255 bytes, each component of the name (for example, Servername, DBname, Ownername, and RPCname) remains limited to 255 bytes.

Should Open Server receive RPC name components with lengths longer than 255 bytes, you receive an error and Open Server copies the first 255 bytes of each component.

Updated Server-Library routines

The following updates have been made to the Server-Library routines:

1 Server properties SRV_S_LOGFILE and SRV_S_TRUNCATELOG can now be set after calling srv_init.

2 After srv_init is called, setting the SRV_S_LOGFILE property with *bufp* set to an empty string ("") and *buflen* set to zero will close the log file.

MIT Kerberos on DB-Library

The MIT Kerberos security mechanism is now available on DB-Library, providing network and mutual authentication services. This feature allows older Sybase applications to use Kerberos authentication services, with less need for modification and recompilation.

The following DB-Library macros were added to enable Kerberos support:

- DBSETLNETWORKAUTH – used to enable or disable network base authentication.
- DBSETLMUTUALAUTH – used to enable or disable mutual authentication of the connection's security mechanism.
- DBSETLSERVERPRINCIPAL – used to set the server's principal name, if required.

Note DB-Library only supports network authentication and mutual authentication services in the Kerberos security mechanism.

❖ **To install MIT-Kerberos on DB-Library**

The following steps provide basic information on installing MIT Kerberos on DB-Library. For more detailed information, refer to *Installation and Release Bulletin* Sybase SDK DB-Lib Kerberos Authentication Option 15.0.

1 Purchase Sybase SDK DB-Lib Kerberos Authentication Option 15.0.

2 Install Sybase SDK DB-Lib Kerberos Authentication Option 15.0 over SDK 15.0 ESD #3 or later.

3 In DB-Library, include *sybdbn.h* instead of *sybdb.h*.

4 Using dbsetversion, set the DB-Library version to DBVERSION_100 or above.

5 Call one or more of the following APIs:

DBSETLNETWORKAUTH(Loginrec *_loginrec_, DBBOOl _enable_)

DBSETLMUTUALAUTH(Loginrec *_loginrec_, DBBOOl _enable_)

DBSETLSERVERPRINCIPAL(Loginrec *_loginrec_, char *_name_)

6 Recompile DB-Library.

JDBC 3.0 features support in jConnect 6.05

This section describes the JDBC 3.0 features that are supported in the current release of jConnect 6.05.

Savepoint support

Adds the Savepoint interface, which contains new methods to set, release, or roll back a transaction to designated savepoints.

Using Savepoints in your transactions

The transaction support in JDBC 2.0 allowed you to have control over a transaction and roll back every change in a transaction. In JDBC 3.0, you are given more control with savepoints: the Savepoint interface allows you to partition a transaction into logical breakpoints, providing control over how much of the transaction gets rolled back.

Setting and rolling back to a Savepoint

The JDBC 3.0 API adds the method Connection.setSavepoint, which sets a savepoint within the current transaction and returns a Savepoint object. The Connection.rollback method is overloaded to take a Savepoint object argument.

Releasing a Savepoint

The Connection.releaseSavepoint method takes a Savepoint object as a parameter and removes it from the current transaction. After a Savepoint has been released, if you try to reference it in a rollback operation, a SQLException occurs.

Any savepoints that you create in a transaction are automatically released and become invalid when the transaction is committed or when the entire transaction is rolled back. If you roll a transaction back to a savepoint, it automatically releases and invalidates any other savepoints that were created after the savepoint in question.

Note You can use the DatabaseMetaData.supportsSavepoints method to determine whether a JDBC API implementation supports savepoints.

Retrieval of parameter metadata

Adds the interface ParameterMetaData, which describes the number, type, and properties of parameters to prepared statements, and supports the new and modified DatabaseMetaData methods.

Retrieval of auto-generated keys

Adds a way to retrieve values from columns that contain automatically generated values. JDBC 3.0 addresses the common need to obtain the value of an auto-generated or auto-incremented key.

Determine the value of a generated key

To inform the driver that you want to retrieve the auto-generated keys, pass the constant Statement.RETURN_GENERATED_KEYS as the second parameter of the Statement.execute() method. After you have executed the statement, call Statement.getGeneratedKeys() to retrieve the generated keys. The result set will contain a row for each generated key retrieved.

Note Adaptive Server cannot return a result set of generated keys. If you execute a batch of insert commands, invoking Statement.getGeneratedKeys() will only return the value of the last generated key.

For more information about retrieving auto-generated keys, including a sample code, search for “retrieving automatically generated keys” on the Sun Microsystems Web site.

Ability to have multiple open ResultSet objects

Adds a new method getMoreResults(int), which takes an argument that specifies whether ResultSet objects returned by a Statement object should be closed before returning any subsequent ResultSet objects.

As a part of the changes, the JDBC 3.0 specification allows the Statement interface to support multiple open ResultSets, which removes the limitation of the JDBC 2 specification that statements returning multiple results must have only one ResultSet open at any given time. To support multiple open results, the Statement interface adds an overloaded version of the method getMoreResults(). The new form of the method takes an integer flag that specifies the behavior of previously opened ResultSets when the getResultSet() method is called. The interface defines the flags as follows:

- CLOSE_ALL_RESULTS – all previously opened ResultSet objects are closed when calling getMoreResults().

- CLOSE_CURRENT_RESULT – the current ResultSet object are closed when calling getMoreResults().
- KEEP_CURRENT_RESULT – the current ResultSet object is not closed when calling getMoreResults().

Passing parameters to *CallableStatement* objects by name

Adds methods to allow a string to identify the parameter to be set for a CallableStatement object.

The CallableStatement interface has been updated to allow you to specify parameters by their names and not the previous method of specifying the parameter's index. You will find this useful when a procedure has many parameters with default values. You can use named parameters to specify only the values that have no default value.

Holdable cursor support

Adds the ability to specify the holdability of a ResultSet object. A holdable cursor, or result, is one that does not automatically close when the transaction that contains the cursor is committed. JDBC 3.0 adds support for specifying cursor holdability. For you to specify the holdability of your ResultSet, you must do so when you prepare a statement using the createStatement(), prepareStatement(), or prepareCall() methods. The holdability may be one of the following constants:

- HOLD_CURSORS_OVER_COMMIT – ResultSet objects (cursors) are not closed; they are held open when a commit operation is implicitly or explicitly performed.
- CLOSE_CURSORS_AT_COMMIT – ResultSet objects (cursors) are closed when a commit operation is implicitly or explicitly performed.

If you close a cursor when a transaction is committed, it usually results in better performance. Unless you require the cursor after the transaction, it is recommended that you close the cursor when the commit operation is carried out. Because the specification does not define the default holdability of a ResultSet, its behavior will depend on the implementation.

New property GET_COLUMN_LABEL_FOR_NAME

Starting with jConnect 6.0, calling the getColumnName method returns the real column name and not the column label. To enable the old behavior, set GET_COLUMN_LABEL_FOR_NAME to True.

New features in ESD #4

This section describes features that are new for ESD #4.

SSL Plus support on Linux AMD64 (Opteron)/EM64T and HP Itanium 32-bit and 64-bit, Sun Solaris 10 x64 32-bit and 64-bit

Sybase now supports SSL Plus on the following platforms:

- SSL5.0.4m on Linux AMD64 (Opteron)/EM64T
- SSL5.0.6f on HP Itanium 32-bit
- SSL5.9.6h on HP Itanium 64-bit
- SSL5.0.4 on Sun Solaris 10 x64 32-bit and 64-bit

The new SSL Plus support applies to both SDK and Open Server.

Extended support for MIT Kerberos

MIT Kerberos 5 version 1.4.1 is now supported on HP Itanium 32-bit and 64-bit.

Table 7 lists releases of MIT Kerberos version 5 on platforms supported by Sybase.

Table 7: MIT Kerberos version 5 releases and supported platforms

MIT Kerberos version 5	Platform
Release 2.6.5	• Windows x86 32-bit (2000 Service Pack 4) • Windows x86 32-bit (2003 Service Pack 1) • Windows x86 32-bit (XP Service Pack 2)
Release 1.4.3	• HP-UX 11.11 64-bit • IBM RS/6000 AIX 5.2 64-bit • Linux AMD64 (Opteron)/EM64T
Release 1.4.2	• Sun Solaris 10 x64 32-bit and 64-bit
Release 1.4.1	• HP-UX 11.11 32-bit • IBM RS/6000 AIX 5.2 32-bit • Linux on POWER 32-bit and 64-bit • Sun Solaris 8 (SPARC) 32-bit and 64-bit • Linux x86 32-bit • HP Itanium 32-bit and 64-bit

To use and configure MIT Kerberos security services for the above platforms, refer to the following documents:

- The chapter and appendixes in the Open Client and Open Server *Configuration Guide for UNIX*:
  - Chapter 6, “Using Security Services”
  - Appendix B, “Configuration Files”
  - Appendix D, “Kerberos Security Services”
- The chapter and appendix in the Open Client and Open Server *Programmer’s Supplement for UNIX*:
  - Chapter 1, “Open Client Client-Library/C”
  - Appendix B, “Environment Variables”
- The chapter and appendix in the Open Client and Open Server *Configuration Guide for Windows*:
  - Chapter 6, “Using Security Services”
  - Appendix B, “Configuration Files”
- The chapters in the Open Client and Open Server *Programmer’s Supplement for Windows*:
  - Chapter 1, “Building Open Client and Open Server Applications”

- Chapter 2, “Client-Library/C Example Programs”

ESQL/COBOL support on HP Itanium 32-bit

ESQL/COBOL is now supported on HP Itanium 32-bit. The COBOL compiler for HP Itanium 32-bit is Micro Focus Server Express 4.0 SP2.

Login redirection and extended HA failover support on Open Server

Login redirection and extended HA failover support allows a cluster of servers to perform load-balancing for all incoming client connections.

To support the new functionality, three new API routines were created, srv_send_ctlinfo, srv_getserverbyname, and srv_freeserveraddrs.

srv_send_ctlinfo was added for both login redirection and extended HA failover support, while srv_getserverbyname, and srv_freeserveraddrs were added to allow an Open Server application to translate a given server name to its connection information. These routines are described in the sections that follow.

The following properties were added to support the new routines:

- SRV_S_HASERVER, a read-only server property that returns the HAFAILOVER value from the interfaces file, which corresponds to the server name as set by srv_init.
- SRV_T_REDIRECT, a read-only thread property that returns the setting of the TDS_HA_LOG_REDIRECT bit in the login record.
- SRV_T_HA, a thread property that returns the setting of HA-related information from the login record as a CS_INT bitmask. Information provided includes session (SRV_HA_LOGIN), failover (SRV_HA_LOGIN_FAILOVER), and resume (SRV_HA_LOGIN_RESUME) bits.
- CS_SESSIONID, a type definition that holds the Session ID.
- SRV_T_SESSIONID, a read-only thread property that returns the Session ID that the client sends to Open Server in the login record.
- SRV_NEG_SESSIONID, a type of negotiated login information in the parlance of srv_negotiate that supports the sending of client Session ID information.

srv_send_ctlinfo

Description

Sends control messages to Client-Library.

Syntax

CS_RETCODE **srv_send_ctlinfo**(SRV_PROC **srvproc*, CS_INT *ctl_type*, CS_INT *paramcnt*, SRV_CTLITEM **param*)

Parameters

ctl_type

An enumerated type with the following values:

- SRV_CTL_LOGINREDIRECT

  Only valid during a connect handler. When used, a SRV_PROC for which SRV_T_REDIRECT is true will instruct the client to restart login using the passed-in server addresses.

- SRV_CTL_HAUPDATE

  Valid at any time srv_sendinfo is valid. When the server sends this message to a client, the client will replace its current HA failover target information with the server connection information as expressed via *param*.

paramcnt

The number of elements in the *param* array

param

Added to pass library control message parameters. It has the following fields:

- SRV_CTLTYPES *srv_ctlitemtype*, where *srv_ctlitemtype* indicates the parameter type. The following types are available:
  - *SRV_CT_SERVERNAME*, which indicates that *srv_ctlptr* points to a CS_CHAR string containing the name of the server whose address will be looked up.
  - *SRV_CT_TRANADDR*, which indicates that *srv_ctlptr* points to a CS_TRANADDR structure containing the server address information.
  - *SRV_CT_ADDRSTR*, which indicates that *srv_ctlptr* points to a string formated by srv_getserverbyname.
  - *SRV_CT_OPTION*, which indicates that *srv_ctlptr* points to a *CS_INT* bitmask that contains a set of options for this message.
- CS_INT *srv_ctllength*, which is the length of variable size parameters. This is currently *SRV_CT_SERVERNAME* and *SRV_CT_ADDRSTR*, where *CS_NULLTERM* is a valid length.

- void **srv_ctlptr*, where *srv_ctlptr* points to the actual parameter data.

srv_getserverbyname

Description

Returns the connection information for *server_name*, allocating memory as needed. *server_name* is the name of the server to be looked up. Memory allocated by srv_getserverbyname can be freed by calling srv_freeserveraddrs.

Syntax

CS_RETCODE **srv_getserverbyname**(CS_CHAR **server_name*, CS_INT *namelen*, CS_INT *querytype*, CS_INT *result_type*, void **resultptr*, CS_INT **result_cnt*)

Parameters

namelen

Length of *server_name*. Can be specified as CS_NULLTERM.

querytype

Allows master (CS_ACCESS_CLIENT_MASTER) or query (CS_ACCESS_CLIENT_QUERY) entries for a given server name.

result_type

Indicates the data format the information will be returned as. Can be specified as SRV_C_GETADDRS, where the information will be returned as an array of CS_TRANADDR structures. Alternatively, you can specify *result_type* as SRV_C_GETSTRS, which returns an array of points to character strings in the *network-protocol protocol-address filter-information* format. For example, where *network-protocol* is “tcp”, *protocol-address* is “myhost 4000”, and *filter-information* is “ssl”, you will receive a result of “tcp myhost 4000 ssl”.

resultptr

A pointer allocated by srv_getserverbyname to hold the results of a query.

result_cnt

A pointer to CS_INT that contains the number of addresses returned for *server_name*.

srv_freeserveraddrs

Description

Frees memory allocated by srv_getserverbyname.

Syntax

CS_RETCODE **srv_freeserveraddrs**(void **resultptr*)

Extended support for OpenLDAP

OpenLDAP is now supported on Linux AMD64 (Opteron)/EM64T and Linux on POWER 32-bit and 64-bit.

You can use the LDAP directory service to create, modify, and retrieve information from network entities.

To enable LDAP, modify the *libtcl.cfg* (32-bit) or *libtcl64.cfg* (64-bit) configuration files, both available in the *$SYBASE/$SYBASE_OCS/config* directory. Then, add an LDAP server to the configured directory service.

Further configuration instructions are provided in Chapter 5, "Using a Directory Service," of the Open Client and Open Server *Configuration Guide for UNIX*.

Asynchronous execution for ODBC

By default, drivers execute ODBC functions synchronously. That is, the application calls a function and the driver returns control to the application when execution is complete. With asynchronous execution, the driver returns control to the application after minimal processing and before execution is complete. This allows the application to execute in parallel other functions while the first function is still executing. Asynchronous execution is beneficial when a task is complex and requires a significant amount of time to execute.

For more information on asynchronous execution and its application, refer to MSDN ODBC Programmer's Reference at http://msdn2.microsoft.com/en-us/library/ms713563.aspx.

Note The ASE ODBC Driver by Sybase supports a maximum of one concurrent statement in asynchronous mode. Only one concurrent statement, synchronous or asynchronous, can be executed if server-side cursors are used or if the connection's auto-commit is disabled.

To use connection-level asynchronous execution with the ASE ODBC Driver by Sybase, call SQLSetConnectAttr and set *SQL_ATTR_ASYNC_ENABLE* to SQL_ASYNC_ENABLE_ON.

Note Calling SQLCancel when no processing is being done will not close the associated cursors. ODBC applications should explicitly call SQLFreeStmt or SQLCloseCursor to close cursors.

Documentation updates and clarifications to ESD #1

Documentation for the “Directory services sql.ini and interface file support” feature in ESD #1 has been updated. For more information, see “Directory services sql.ini and interface file support” on page 130.

New features in ESD #3

This section describes features that are new for ESD #3.

ESQL/COBOL support for scrollable cursors

ESQL/COBOL now supports scrollable cursors, which allow you to set the current cursor position anywhere in the result set by specifying the fetch orientation. Both single row fetches and multiple row fetches are supported. By default, if the row count is not set at cursor open time, a fetch will return only one row. This default behavior is illustrated in COBOL samples *example3.pco* and *example4.pco*.

Scrollable cursors are read-only with INSENSITIVE or SEMI_SENSITIVE properties. Two new samples are included for the feature: *example3.cp* shows usage of an INSENSITIVE scrollable cursor; *example4.cp* shows usage of a SEMI_SENSITIVE scrollable cursor.

Note For more information on scrollable cursor support, refer to “Open Server support for scrollable cursors” on page 118, “Embedded SQL/C scrollable cursors” on page 125, and “Scrollable cursors supported” on page 152.

The following sections describe ESQL/COBOL statements that have changed due to the introduction of scrollable cursors.

DECLARE statement

Description — Includes syntax added for scrollable cursors.

Syntax

```
EXEC SQL DECLARE <curs_name>
  [ <cursor sensitivity> ]
  [ <cursor scrollability> ] CURSOR
```

```
    FOR <cursor specification>
        <cursor sensitivity> : : =
    SEMI_SENSITIVE
    | INSENSITIVE
  <cursor scrollability> : : =
    SCROLL
    | NO SCROLL
  <cursor specification> : : =
  <select statement> [ <updatability clause> ]
  <updatability clause> : : =
  FOR {READ ONLY | UPDATE [ OF <column name list> ]}
  END-EXEC
```

Parameters

cursor sensitivity

Specified as SEMI_SENSITIVE or INSENSITIVE:

- If *cursor sensitivity* is specified as INSENSITIVE, SCROLL is not implied.
- If *cursor sensitivity* is not specified as INSENSITIVE or SEMI_SENSITIVE, and SCROLL is also not specified in the declare cursor, the cursor is scrollable and read-only with the specified sensitivity.
- If *cursor sensitivity* is not specified, the cursor is declared as non-sensitive, non-scrollable and read-only.

cursor scrollability

Specified as SCROLL or NO SCROLL:

- If *cursor scrollability* is specified as SCROLL, the cursor is INSENSITIVE.
- If *cursor scrollability* is not specified, the default is NO SCROLL, and the cursor is declared as non-scrollable and read-only.

Note A scrollable cursor does not use fetch loops but rather single fetch calls. Only non-scrollable and forward-only cursors use fetch loops.

OPEN statement

Description

Includes syntax added for row pre-fetching.

Syntax

EXEC SQL OPEN <cursor_name> [*ROW_COUNT* = *size*] END-EXEC

Parameters

size

Specified as the pre-fetch count. The value is the same as the host array size.

ROW_COUNT

Specified only when host arrays are used as host variables.

FETCH statement

Description

Fetches single or multiple rows from the cursor result set, depending on the ROW_COUNT specification at CURSOR OPEN time.

If a cursor is specified as scrollable, the *fetch orientation* in the FETCH statement specifies the fetch direction.

If the cursor is not specified as scrollable, FETCH retrieves the next row in the result set.

Syntax

```
EXEC SQL DECLARE [ <fetch orientation> ]
        [ FROM ] <cursor name>
        { [ INTO <fetch target list> ] |
        [SQL DESCRIPTOR <>]
<fetch orientation> : : =
      | NEXT
      | PRIOR
      | FIRST
      | LAST
      | ABSOLUTE <fetch_offset>
      | RELATIVE <fetch_offset>
<fetch offset> : : =
      <signed_numeric_literal>
<fetch target list> : : =
      <target specification>
      [ { <comma> <target specification> } ]
END-EXEC
```

Parameters

fetch orientation

Specified as NEXT, PRIOR, FIRST, LAST, ABSOLUTE, or RELATIVE.

If *fetch orientation* is not specified, NEXT is the default.

Note If you specify *fetch orientation* as any type except NEXT on a non-scrollable cursor, you receive the following message:

```
The fetch type can only be used with scrollable cursors.
```

If *fetch orientation* positions the cursor beyond the last row or before the first row, *sqlca.sqlcode* is set to 100, indicating that no rows are found. If an error handler is installed, it may provide additional information.

---

fetch offset

Specified as an exact, signed numeric value with a scale of zero.

Examples

The following syntax examples declare an INSENSITIVE scrollable cursor and fetch rows at random.

To declare an INSENSITIVE scrollable cursor and fetch rows at random:

```
EXEC SQL DECLARE c1 INSENSITIVE SCROLL CURSOR FOR
  select title_id, royalty
  from authors
  where royalty < 25 END-EXEC.
EXEC SQL OPEN c1 END-EXEC.
```

To fetch a row when a cursor is declared and open:

```
EXEC SQL FETCH LAST FROM c1 INTO :title,:roy END-EXEC.
```

To fetch a previous row:

```
EXEC SQL FETCH PRIOR FROM c1 INTO :title,:roy END-EXEC.
```

To fetch row 20:

```
EXEC SQL FETCH ABSOLUTE 20 FROM c1 INTO :title, :roy
END-EXEC.
```

Extended support for MIT Kerberos

MIT Kerberos 5 on HP-UX 11.11 64-bit and IBM RS/6000 AIX 5.2 64-bit is rebased from version 1.3.6 to 1.4.3.

MIT Kerberos 5 version 1.4.1 is now supported on HP-UX 11.11 32-bit, IBM RS/6000 AIX 5.2 32-bit, Linux on POWER 32-bit and 64-bit.

Table 8 lists releases of MIT Kerberos version 5 on platforms currently supported by Sybase.

Table 8: MIT Kerberos version 5 releases and supported platforms

MIT Kerberos version 5	Platform
Release 2.6.5	• Windows x86 32-bit (2000 Service Pack 4) • Windows x86 32-bit (2003 Service Pack 1) • Windows x86 32-bit (XP Service Pack 2)
Release 1.4.3	• HP-UX 11.11 64-bit • IBM RS/6000 AIX 5.2 64-bit
Release 1.4.1	• HP-UX 11.11 32-bit • IBM RS/6000 AIX 5.2 32-bit • Linux on POWER 32-bit and 64-bit • Sun Solaris 8 (SPARC) 32-bit and 64-bit • Linux x86 32-bit

To use and configure MIT Kerberos security services for the platforms in Table 8, refer to the following documents:

- The chapter and appendixes in the Open Client and Open Server *Configuration Guide for UNIX*:
  - Chapter 6, “Using Security Services”
  - Appendix B, “Configuration Files”
  - Appendix D, “Kerberos Security Services”
- The chapter and appendix in the Open Client and Open Server *Programmer’s Supplement for UNIX*:
  - Chapter 1, “Open Client Client-Library/C”
  - Appendix B, “Environment Variables”
- The chapter and appendix in the Open Client and Open Server *Configuration Guide for Windows*:
  - Chapter 6, “Using Security Services”
  - Appendix B, “Configuration Files”
- The chapters in the Open Client and Open Server *Programmer’s Supplement for Windows*:

- Chapter 1, “Building Open Client and Open Server Applications”
- Chapter 2, “Client-Library/C Example Programs”

ESQL/COBOL support on Linux x86 32-bit

ESQL/COBOL support for Linux x86 32-bit on Redhat AS EL 3.0 is now available. The COBOL compiler for Linux x86 32-bit is Micro Focus Server Express 4.0 SP2.

HA failover on ASE OLE DB Provider

Table 9 lists the new connection parameters used for High Availability (HA) failover support on the Adaptive Server Enterprise (ASE) OLE DB Provider:

Table 9: HA failover connection parameters

Property names	Description	Required	Default value
HASession	Specifies if high availability is enabled. 0 indicates high availability disabled, 1 high availability enabled.	No	0
SecondaryPort	The port number of the ASE server acting as a failover server in an active-active or active-passive setup.	Yes, if HASession is set to 1.	Empty
SecondaryServer	The name or the IP address of the ASE server acting as a failover server in an active-active or active-passive setup.	Yes, if HASession is set to 1.	Empty

Using failover in HA systems

Description of the failover process

A High Availability (HA) cluster is used in an environment that must always be available, such as a banking system to which clients must be able to connect, 365 days a year.

A HA cluster includes two or more machines that are configured so that if one machine (or application) is interrupted, the second machine assumes the workload of both machines. Each machine is one node of the High Availability cluster.

The machines are configured so that each machine can read the other machine's disks, although not at the same time. All of the disks that are failed-over should be shared disks.

For example, if Adaptive Server 1 is the primary companion server, and it crashes, Adaptive Server 2, as the secondary companion server, reads its disks (disks 1 - 4) and manages any databases on them until Adaptive Server 1 can be rebooted. Any clients connected to Adaptive Server 1 are automatically connected to Adaptive Server 2.

Failover allows Adaptive Server to work in a HA cluster in active-active or active-passive configuration.

Connection properties

During failover, clients connected to the primary companion using the failover property automatically reestablish their network connections to the secondary companion. Failover can be enabled by setting the connection property HASession to "1" (default value is "0"). If this property is not set, the session failover does not occur, even if the server is configured for failover. Also, you must set SecondaryServer (the IP address or the machine name of the secondary ASE server) and SecondaryPort (the port number of the secondary ASE server) properties.

When the OLE DB Provider driver detects a connection failure with the primary ASE server, it first tries to reconnect to the primary. If it cannot reconnect, it assumes that a failover has occurred. Then, it automatically tries to connect to the secondary ASE server using the connection properties set in SecondaryServer and SecondaryPort.

See the ASE book, *Using Sybase Failover in a High Availability System*, for information about configuring your system for HA.

Confirming a successful failover

If a connection to the secondary ASE server is established, the ASE OLE DB Driver returns "E_FAIL" for the function return HRESULT.

To confirm a successful failover, do one of the following:

- Examine the dwMinor field in ERRORINFO (returned from IErrorRecords::GetBasicErrorInfo), or the description returned from IErrorInfo::GetDescription. The dwMinor value should be “30130” for a successful HA failover.
- Examine the description from IErrorInfo::GetDescription. The description should be as follows, where *ASEServerName* is the server name failed over to:

  ```
  “Sybase server is not available or has terminated
  your connection, you have successfully connected to
  the next available HA server ASEServerName. All
  transactions has been rolled back.”
  ```

Note Sybase recommends that you check the code returned by dwMinor to determine the success of the failover rather than by examining the error description.

Then the client must then reapply the failed transaction with the new connection. If failover happens while a transaction is open, only the changes that were committed to the database before failover are retained.

Verifying an unsuccessful failover

If the connection to the secondary server is not established, the ASE OLE DB Driver also returns “E_FAIL” for the function return HRESULT. However, the dwMinor field in ERRORINFO (returned from IErrorRecords::GetBasicErrorInfo) should be “30131”, and the description returned from IErrorInfo::GetDescription should be:

```
“Connection to Sybase server has been lost, connection
to the next available HA server also failed. All
transactions have been rolled back.”
```

Sample code for checking failover

The following code snippet shows how to code for a failover:

```
/* Declare required variables */
...
/* Open Database connection */
...
/* Perform a transaction */
...
/*Check HRESULT  and dwMinor in ERRORINFO, handle
   failover */
if (FAILED(hr))
```

```
{
   IErrorInfo* pIErrorInfo;
   GetErrorInfo(0, &pIErrorInfo);
   IErrorRecords * pIErrorRecords;
   HRESULT hr1 = pIErrorInfo->QueryInterface(
     IID_IErrorRecords,(void **)&pIErrorRecords);

   if (SUCCEEDED(hr1))
   {
      ERRORINFO errorInfo;
     pIErrorRecords->GetBasicErrorInfo(0,
        &errorInfo);
     pIErrorRecords->Release();
     if (errorInfo.dwMinor == 30130)
     {
        //successful failover,
        //retry the transaction
      }
   }
}
```

New jConnect 6.05 connection properties

Table 10 lists new connection properties available with ESD #3. For a complete list of connection properties, refer to Chapter 2, "Programming Information," in the jConnect for JDBC 6.05 *Programmer's Reference*.

Table 10: Connection properties for jConnect 6.05

Property	Description	Default value
CRC	When this property is set to *true*, the update counts that are returned are cumulative counts that include updates directly affected by the statement executed and any triggers invoked as a result of the statement being executed.	*false*
DATABASE	Use this property to specify the database name for a connection when the connection information is obtained from a Sybase *interfaces* file. The URL of an *interfaces* file cannot supply the database name.	*null*
DEFAULT_QUERY_TIMEOUT	When this connection property is set, it is used as the default query timeout for any statements created on this connection.	*0* (no timeout)

Property	Description	Default value
IMPLICIT_CURSOR_FETCH_SIZE	Use this property in conjunction with the SELECT_OPENS_CURSOR property to force jConnect to open a read-only cursor on every select query that is sent to the database. The cursor will have a fetch size of the value set in this property, unless overridden by the Statement.setFetchSize method.	*0*
INTERNAL_QUERY_TIMEOUT	Use this property to set the query timeout that will be used by statements internally created and executed by jConnect. This may prevent application hangs if internal commands do not complete in a reasonable time.	*0* (no timeout)
J2EE_TCK_COMPLIANT	When this property is set to *true* ,the jConnect driver enables behaviour that is compliant with the J2EE 1.4 TCK test suite—this causes some loss of performance. Therefore, Sybase recommended using the default value of *false*.	*false*

New features in ESD #2

This section describes features that are new for ESD #2.

Open Server support for scrollable cursors

The scrollable cursor feature is now available on Open Server.

A new capability, CS_REQ_CURINFO3, is added to Open Server to support the new scrollable cursor feature. During login, CS_REQ_CURINFO3 allows a remote client connecting to Open Server to request scrollable cursor support.

Scrollable cursors on Open Server are read-only with INSENSITIVE or SEMI_SENSITIVE properties. Non-scrollable, insensitive cursors are also supported on Open Server, and are set with the CS_NOSCROLL_INSENSITIVE option.

Note If you use the CTOS application, do not use the ct_scroll_fetch routine with non-scrollable cursors. Instead, use the ct_fetch routine.

New SRV_CURDESC2 scrollable cursor structure

The new SRV_CURDESC2 scrollable cursor structure in Open Server is a superset of the SRV_CURDESC cursor structure.

There are two additional fields in the SRV_CURDESC2 structure:

- *currow_pos* describes the current row position of a cursor.
- *curtotalrowcount* describes the total number of rows in the result set; *curtotalrowcount* only applies to insensitive, scrollable cursors.

The following new cursor declare options are available in the *curstatus* field in SRV_CURDESC2:

- CS_CURSTAT_SCROLLABLE specifies a read-only, insensitive scrollable cursor.
- CS_CURSTAT_INSENSITIVE specifies a read-only, non-scrollable, insensitive cursor. When such a cursor is specified, CS_CURSTAT_INSENSITIVE must be enabled, and CS_CURSTAT_SCROLLABLE must be disabled.

  When an insensitive, scrollable cursor is specified, both CS_CURSTAT_INSENSITIVE and CS_CURSTAT_SCROLLABLE must be enabled.
- CS_CURSTAT_SEMISENSITIVE specifies a read-only, semi-sensitive, scrollable cursor. When such a cursor is specified, CS_CURSTAT_SCROLLABLE must also be enabled.

New *srv_cursor_props2* routine added

The srv_cursor_props2 routine is added to Open Server to support the new SRV_CURDESC2 structure.

For pre-15.0 applications, you must use the SRV_CURDESC structure and srv_cursor_props routine, if the application sets CS_VERSION_125.

For 15.0 applications that support scrollable cursors on Open Server, use the SRV_CURDESC2 structure, and set the application to CS_VERSION_150.

The arguments for srv_cursor_props2 are as follows:

```
ret = srv_cursor_props2(SRV_PROC *spp, CS_INT cmd,
SRV_CURDESC2 *cdp);
```

New scrollable cursor command options

New scrollable cursor command options (cmdoptions) for Open Server are described in Table 11, together with Open Client equivalents.

Table 11: Open Client/Server scrollable cursor command options

Open Client command options	Open Server command options
CS_CUR_SCROLL	SRV_CUR_SCROLL
CS_CUR_SCROLL_INSENS	SRV_CUR_SCROLL_INSENS
CS_CUR_SCROLL_SEMISENS	SRV_CUR_SCROLL_SEMISENS
CS_CUR_NOSCROLL_INSENS	SRV_CUR_NOSCROLL_INSENS

These cmdoptions are valid only at the cursor declare cycle, where the *curcmd* field of the SRV_CURDESC2 structure may contain one of these options, based on the remote client issuing a ct_cursor (CS_CURSOR_DECLARE) command.

Extended support for MIT Kerberos

Table 12 lists releases of MIT Kerberos version 5 on platforms supported by Sybase.

Table 12: MIT Kerberos version 5 releases and supported platforms

MIT Kerberos version 5	Platform
Release 2.6.5	• Windows x86 32-bit (2000 Service Pack 4) • Windows x86 32-bit (2003 Service Pack 1) • Windows x86 32-bit (XP Service Pack 2)
Release 1.4.1	• Sun Solaris 8 (SPARC) 32-bit and 64-bit • Linux x86 32-bit
Release 1.3.6	• HP-UX 11.11 64-bit • IBM RS/6000 AIX 5.2 64-bit

To use and configure MIT Kerberos security services for the above platforms, refer to the following documents:

- The chapter and appendixes in the Open Client and Open Server *Configuration Guide for UNIX*:
  - Chapter 6, “Using Security Services”
  - Appendix B, “Configuration Files”
  - Appendix D, “Kerberos Security Services”

- The chapter and appendix in the Open Client and Open Server *Programmer's Supplement for UNIX*:
  - Chapter 1, "Open Client Client-Library/C"
  - Appendix B, "Environment Variables"
- The chapter and appendix in the Open Client and Open Server *Configuration Guide for Windows*:
  - Chapter 6, "Using Security Services"
  - Appendix B, "Configuration Files"
- The chapters in the Open Client and Open Server *Programmer's Supplement for Windows*:
  - Chapter 1, "Building Open Client and Open Server Applications"
  - Chapter 2, "Client-Library/C Example Programs"

Extended support for OpenLDAP

OpenLDAP (LDAP) 2.2.26 is now supported by Sybase on the following platforms:

- HP-UX 11.11 32-bit and 64-bit
- IBM RS/6000 AIX 5.2 32-bit and 64-bit
- Linux x86 32-bit
- Sun Solaris 8 (SPARC) 32-bit and 64-bit

Prior to ESD #2, these platforms used Netscape LDAP 4.0, Netscape LDAP 4.1, or OpenLDAP 2.2.3.

You can use the LDAP directory service to create, modify, and retrieve information from network entities.

To enable LDAP, modify the *libtcl.cfg* (32-bit) or *libtcl64.cfg* (64-bit) configuration files, both available in the *$SYBASE/$SYBASE_OCS/config* directory. Then, add an LDAP server to the configured directory service.

Further configuration instructions are provided in Chapter 5, "Using a Directory Service," of the Open Client and Open Server *Configuration Guide for UNIX*.

ASE OLE DB Provider participation in Distributed Transactions

This feature is supported only on Windows and requires that Microsoft Distributed Transaction Coordinator (MS DTC) be the transaction coordinator managing Distributed Transactions.

Sybase supports the following programming models:

- Applications using MS DTC directly.
- Applications using Microsoft Transaction Server (MTS) or (COM+).

Programming for MS DTC

❖ **To program using Microsoft Distributed Transaction Coordinator (MS DTC)**

1 Connect to MS DTC using the DtcGetTransactionManager function. For information about MS DTC, see Microsoft Distributed Transaction Coordinator documentation.

2 Get the IDBSession for each Sybase ASE connection you want to establish following the OLE DB steps.

3 Call the ITransactionDispenser::BeginTransaction function to begin an MS DTC transaction and to obtain an OLE Transaction object that represents the transaction.

4 Query ITransactionJoin from each IDBSession (OLE DB Connection) you want to enlist in the MS DTC transaction, and call JoinTransaction with the passed in parameter punkTransactionCoord as the Transaction object (obtained in Step 3).Currently, Sybase supports only the isolation level of ISOLATION LEVEL_READCOMMITTED for the distributed transaction, and does not support ITransactionOptions.

5 To update a SQL Server, follow the OLE DB steps for creating and executing the IDBCommand.

6 Call the ITransaction::Commit function to commit the MS DTC transaction. The Transaction object is no longer valid.

Programming components deployed in MTS or COM+

The following procedure describes how to create components that participate in Distributed Transactions in MTS or COM+.

❖ **To program components deployed in MTS or COM+**

1 Create an IDBSession for each ASE connection.

2 Create and execute IDBCommand for each update you would like to perform.

3 Deploy your component to MTS or COM, and configure the transaction attributes as needed.

The COM+, OLE DB Services, and OLE DB provider will take care of creating the transaction, participating in the transaction, and committing or rolling back the transaction.

OLE DB Services is needed for the Automatic Transaction Enlistment. To enable the OLE DB Services, you must follow some rules to initialize the Data Source (see the MS OLE DB documents). To enable the automatic transaction enlistment, you can set the bit DBPROPVAL_OS_TXNENLISTMENT in the *OLE_DB_SERVICES* registry and in the DBPROP_INIT_OLEDBSERVICES property value, or pass OLE DB Services = *2* in the connection string.

Connection properties for Distributed Transaction support

The following describes the connection properties:

- Distributed Transaction Protocol (DistributedTransactionProtocol) – To specify the protocol used to support the distributed transaction, either use the XA Interface standard or MS DTC OLE Native protocol, select the Distributed Transaction Protocol in the OLE DB Data Source Dialog, set the property DistributedTransactionProtocol = *OLE* in the provider string part of the connection string for OLE Native protocol, or the use default protocol *XA*.
- Tightly Coupled Transaction (TightlyCoupledTransaction) – When a distributed transaction using two resource managers points to the same ASE server, you may have a situation called “Tightly Coupled Transaction.” Under these conditions, if you do not set this property to 1, the Distributed Transaction may fail.

To summarize, if you open two database connections to the same ASE server and then enlist these connections in the same distributed transaction, Sybase recommends that you set TightlyCoupledTransaction=1. To set this property, select the Tightly Coupled Transaction in the OLE DB Data Source dialog box, or pass the property TightlyCoupledTransaction=1 in the provider string part of the connection string.

OLE DB DSN Migration tool available

The OLE DB DSN Migration tool helps you to migrate your Data Source Names (DSNs) from the OLE DB Driver Kit to the OLE DB Provider by Sybase. When you migrate the DSNs, they will use the new OLE DB Provider by Sybase instead of the OLE DB Driver Kit.

For information on migrating your applications from the OLE DB Driver Kit to the ASE OLE DB Provider by Sybase, see “Migrating to ASE OLE DB Provider by Sybase” on page 147.

There are two methods to migrate DSNs from the OLE DB Driver Kit to the drivers created by Sybase:

- Using the Sybase ASE Data Source Administrator
- Using the DSN Migration tool

Using the Sybase ASE Data Source Administrator

The Sybase ASE Data Source Administrator is a GUI process that allows you to migrate existing OLE DB Driver Kit data sources and to create new data sources for the ASE OLE DB Provider.

❖ **To migrate the data sources using the Data Source Administrator**

1 On the main window titled “Sybase Data Source Administrator,” choose the data source.

2 Click Migrate.

The Sybase Data Source Administrator allows you to add, remove, configure, or test the OLE DB data sources.

Using the DSN Migration tool

The DSN Migration tool can help you migrate the data sources from the OLE DB Driver Kit to the OLE DB Driver by Sybase.

The dsnmigrate tool uses switches to control which DSNs are migrated. You need to enter the following from the command line:

```
dsnmigrate.exe [/?|/h|/help][/oledb]
[/l|/ul|/sl][/a|/ua|/sa] [[/dsn|/udsn|/sdsn]=dsn]
[/suffix=suffix]
```

For all OLE DB DSNs that are converted, the new Sybase DSNs will have the same name.

Conversion switches

The following table lists and describes the switches used in the conversion.

Table 13: Conversion switches

Switches	Description of results
/?,/h,/help	Displays this message. This message is also displayed if dsnmigrate is called with no command line arguments.
/oledb	Places dsnmigrate into OLEDB-mode. By default, ODBC DSNs are migrated.
/l	Displays a list of all OLE DB Driver Kit user and system DSNs.
/ul	Displays a list of all OLE DB Driver Kit user DSNs.
/sl	Displays a list of all OLE DB Driver Kit system DSNs.
/a	Converts all OLE DB Driver Kit user and system DSNs.
/ua	Converts all OLE DB Driver Kit user DSNs.
/sa	Converts all OLE DB Driver Kit system DSNs.
/dsn	Converts specific OLE DB Driver Kit user or system DSNs.
/udsn	Converts specific OLE DB Driver Kit user DSNs.
/sdsn	Converts specific OLE DB Driver Kit system DSNs.
dsn	The name of the DSN to be converted.
/suffix	An optional switch that changes the way DSNs are named. If this switch is used, the original DSN is retained and the new DSN is named "<dsn>-<suffix>."
suffix	The suffix that is used to name the new DSN.

New features in ESD #1

The following features are new for ESD #1:

Embedded SQL/C scrollable cursors

The scrollable cursor feature is now available with the ESQL/C pre-processor. Similar to Client-Library implementation of scrollable cursors, ESQL/C application programmers can now set the current position anywhere in the result set by specifying a NEXT, PRIOR, FIRST, LAST, ABSOLUTE or RELATIVE clause in an OFFSET statement. It implements a scrollable cursor that is read-only with either an INSENSITIVE or a SEMI_SENSITIVE property.

Declaring cursors

Declare a cursor for each select statement that returns multiple rows of data. You must declare the cursor before using it, and you cannot declare it within a declare section.

The syntax for declaring a cursor is:

```
exec sql declare cursor_name [cursor sensitivity]
[cursor scrollability] cursor
     for  select_statement ;
```

where:

- *cursor_name* identifies the cursor. The name must be unique and have a maximum of 255 characters. The name must begin with a letter of the alphabet or with the symbols "#" or "_".
- *cursor sensitivity* specifies the sensitivity of the cursor. The options are:
  - semi_sensitive. If semi_sensitive is specified in the declare statement, scrollability is implied. The cursor is semi_sensitive, scrollable, and read-only.
  - insensitive. If insensitive is specified in the declare statement, the cursor is insensitive. Scrollability is determined by specifying SCROLL in the declare part. If SCROLL is omitted or NOSCROLL is specified, the cursor is insensitive only and non-scrollable. It is also read-only.

  If *cursor sensitivity* is not specified, the cursor is non-scrollable and read-only.
- *cursor scrollability* specifies the scrollability of the cursor. The options are:
  - scroll. If scroll is specified in the declare statement and sensitivity is not specified, the cursor is insensitive and scrollable. It is also read-only.
  - no scroll. If the SCROLL option is omitted or NOSCROLL is specified, the cursor is non-scrollable and read-only. See the previous *cursor sensitivity* description for cursor behavior.

  If *cursor scrollability* is not specified, the cursor is non-scrollable and read-only.

- select_statement is a select statement that can return multiple rows of data. The syntax for select is the same as that shown in the *Adaptive Server Enterprise Reference Manual*, except that you cannot use into or compute clauses.

Fetching data

Use a fetch statement to retrieve data through a cursor and assign it to host variables. The syntax for the fetch statement is:

```
exec sql [at connect_name] fetch [fetch
orientation] cursor_name
 into : host_variable
 [[ indicator]: indicator_variable ]
 [,: host_variable
 [[ indicator]: indicator_variable ]...];
```

where one *host_variable* exists for each column in the result rows.

Prefix each host variable with a colon, and separate it from the next host variable with a comma. The host variables listed in the fetch statement must correspond to Adaptive Server values that the select statement retrieves. Thus, the number of variables must match the number of returned values, they must be in the same order, and they must have compatible datatypes.

The *fetch orientation* specifies the fetch direction of the row to be fetched, if a cursor is scrollable. The options are: NEXT, PRIOR, FIRST, LAST, ABSOLUTE *fetch_offset* and RELATIVE *fetch_offset*. If fetch orientation is not specified, next is default. If fetch orientation is specified, the cursor must be scrollable.

The data that the fetch statement retrieves depends on the cursor position. The fetch statement typically retrieves single or multiple rows from the cursor result set, depending on the ROW_COUNT specification at cursor open time. If a cursor is not scrollable, fetch retrieves the next row in the result set. If a cursor is scrollable, commands in the fetch statement specify the row position to be fetched.

Declaring a scrollable cursor and fetching rows

To declare a scrollable cursor and fetch rows at random, specify the scroll sensitivity and scrollability in the declare cursor, then specify the fetch orientation at fetch time.

The following example demonstrates declaring an insensitive scrollable cursor and fetching rows at random:

```
exec sql declare c1 insensitive scroll cursor for
 select title_id, royalty, ytd_sales from authors
 where royalty < 25;
exec sql open c1;
```

In this example, scroll and insensitive are specified in the declare cursor. A fetch orientation can be specified at fetch time to indicate which row is required from the result set.

The following fetch example fetches the specified columns of the first row from the result set:

```
exec sql fetch first from c1 into :title,:roy,:sale;
```

The following fetch example fetches the specified columns of the previous row from the result set:

```
exec sql fetch prior from c1 into :title,:roy,:sale;
```

The following fetch example fetches the specified columns of row twenty from the result set:

```
exec sql fetch absolute 20 from c1 into
:title,:roy,:sale;
```

Use *sqlcode* or *sqlstate* to determine if fetch statements return valid rows. For scrollable cursors, it is possible to fetch 0 rows if the cursor is positioned outside of result set boundaries, for example, before the first row or after the last row. In these circumstances, fetching 0 rows is not an error.

BCP support for encrypted columns

If ASE supports encrypted columns, BCP supports bulk copying of encrypted columns. When encrypted columns is supported, bcp sends the following SQL command: set ciphertext on.

The encrypted data is obtained in cipher text instead of plain text.

bcp syntax changes

A new parameter -C is included to support BCP encrypted columns. -C enables the ciphertext option before initiating the bulk copy operation, if Adaptive Server supports encrypted columns.

Example

In the following example, the -C parameter copies data out of the publishers table (with encrypted columns) in cipher-text format, instead of plain text. Pressing Return accepts 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]:
```

BCP and ISQL support for alternative *trusted.txt* file location

The bcp and isql utilities now allow you to specify an alternative location for the *trusted.txt* file.

A new parameter -x is included in the syntax, giving you the option of specifying an alternative *trusted.txt* file location.

Client-Library migration samples available

DB-Library to Client-Library migration samples are now available in the following OCS installation directories:

- UNIX: *$SYBASE/$SYBASE_OCS/sample/db2ct*
- Windows: *%SYBASE%\%SYBASE_OCS%\sample\db2ct*

Samples provided are identical to those available on the Sybase World Wide Web page at http://www.sybase.com/detail?id=1013159.

The *README* file provided with the migration samples contains a list of samples available and instructions on how to build them.

The *makefile* for all UNIX and Linux platforms is available in *$SYBASE/$SYBASE_OCS/sample/db2ct*.

The *makefile* for Microsoft Windows is available in *%SYBASE%\%SYBASE_OCS%\sample\db2ct*.

Note that the Microsoft Windows *makefile* available on the Web is called "*ms.mak*". As a result, the build instructions on Microsoft Windows, as described in the *README* file, are slightly different.

For further information on the sample migration programs and how to migrate Open Client DB-Library applications to Open Client Client-Library, refer to the Open Client 15.0 *Client-Library Migration Guide*.

Directory services sql.ini and interface file support

This new feature supports the use of the *sql.ini* file (for Windows) and the *interfaces* file (for UNIX) to provide server information. It is supported for the following drivers and providers:

- ADO.NET Data Provider
- ODBC Driver
- OLE DB Provider

Currently, to connect to the driver or provider you must specify several properties, such as server name or IP address and port number of an ASE server. By using the *sql.ini* or *interfaces* file, enterprises can centralize the information about the services available in the enterprise networks including, ASE server information.

Connection string

The following must be added to the connection string to identify the *sql.ini* or *interfaces* file. You can connect to a single Directory Services URL (DSURL) or to multiple DSURLs.

Connection string for a single DSURL

For a single DSURL for ADO.NET, ODBC, and the OLE DB drivers and providers, you must add the following properties to the connection string in the following format:

```
DSURL=file://[path]<filename>[?][servicename]
```

where:

- *path* (optional) is the path to the interfaces file. If not specified, *%SYBASE%\ini* is used as the default path on Windows and *$SYBASE* on Linux and Mac OS X.

- *filename* is the name of the interface file.
  - On Windows, the interfaces file is typically named *sql.ini*.
  - On UNIX, the interfaces file is typically named *interfaces*.
- *servicename* (optional) is the name of the service defined by the DSURL, also known as the server name.
  - If not defined in the DSURL, the server property in the connection string is used.
  - If both service name and server exist, the service name will be used.
  - If neither of them exist, the driver or provider loads the server information from the *sql.ini* or *interfaces* file, if the *sql.ini* or *interfaces* file has only one entry. If more than one entry exists in the file, an error is returned.

    ODBC error example:

```
[Sybase][ODBC Driver]Getting more than one servers with the connect
string.
```

    Therefore the servicename or server property should always be defined in the connection URL, as typically interfaces files will contain multiple server entries.

Examples of the connection string for Windows and UNIX:

Windows

Example using a default path:

```
DSURL=file://sql.ini?mango1
```

where:

- path is omitted in the above example, hence *%SYBASE%/ini* is used as the default path. However, if %SYBASE% is defined as *C:\Sybase*, then *C:\Sybase\ini* will be the path used.
- filename is *sql.ini*.
- servicename is *mango1*.

Example using explicit path:

```
DSURL=file://\\myServer\myShare\sql.ini?mango1
```

where:

- path is specified as *\\myServer\myShare*, which is a Universal Naming Convention (UNC) path that refers to a network shared directory.
- filename is *sql.ini*.
- servicename is *mango1*.

Linux and Mac OS X

Example using a default path:

```
DSURL=file://interfaces?mango1
```

where:

- path is omitted in the above example, hence $SYBASE is used as the default path. However, if $SYBASE is defined as */usr/sybase*, then */usr/sybase* will be the path used.
- filename is *interfaces*.
- servicename is *mango1*.

Example using explicit path:

```
DSURL=file:///remote/sybase/interfaces?mango1
```

where:

- path is specified as */remote/sybase*
- filename is *interfaces*.
- servicename is *mango1*.

Connection string for multiple URLs

DSURL can support multiple LDAP URLs and *interfaces* files. Sybase also supports a multiple URL with mixed LDAP URLs and file URLs. When multiple URLs are used, the driver processes the URLs one by one until it successfully opens an *interfaces* file or connects to an LDAP server.

Examples of multiple URLs

The following are examples of using multiple URLs in a connection string:

```
DSURL={file:///mils1/sybase/sql.ini;file:///test/interface}
DSURL={ldap://SYBLDAP:389/dc=sybase,dc=com??one?sybaseServername=MANGO;
ile:///test/interface?MANGO}
```

Format of the interfaces file for Microsoft Windows, UNIX, and SSL

For information on the format of the interfaces file, see the *Adaptive Server Enterprise Configuration Guide* for your platform.

ODBC DSN Migration tool available

The ODBC DSN Migration tool can help you migrate from the ODBC Driver Kit to the ODBC Driver by Sybase. When you migrate your DSNs, they will begin using the new ODBC Driver by Sybase instead of the ODBC Driver Kit.

For information on migrating your applications from the ODBC Driver Kit to the ASE ODBC Driver by Sybase, see “Migrating to ASE ODBC Driver by Sybase” on page 146.

Using the Migration tool

The dsnmigrate tool uses switches to control which DSNs are migrated. To use the tool, you need to enter the following from the command line:

```
dsnmigrate.exe [/?|/help] [l|/ul|/sl][/a|/ua|/sa]
[[/dsn|/udsn|/sdsn]=dsn] [/suffix=suffix]
```

All DSNs that are converted are renamed to “<*dsn*>-backup” before the conversion is completed. When the new Sybase DSNs are created and the conversion is completed, the name is changed to“<*dsn*>,” which will allow existing applications to continue to run without any modifications.

Conversion switches

The following table lists and describes the switches used in the conversion.

Table 14: Conversion switches

Switches	Description of results
/?,/h,/help	Displays this message. This message is also displayed if dsnmigrate is called with no command line arguments.
/l	Displays a list of all ODBC Driver Kit user and system DSNs.
/ul	Displays a list of all ODBC Driver Kit user DSNs.
/sl	Displays a list of all ODBC Driver Kit system DSNs.
/a	Converts all ODBC Driver Kit user and system DSNs.
/ua	Converts all ODBC Driver Kit user DSNs.
/sa	Converts all ODBC Driver Kit system DSNs.
/dsn	Converts specific ODBC Driver Kit user or system DSNs.
/udsn	Converts specific ODBC Driver Kit user DSNs.
/sdsn	Converts specific ODBC Driver Kit system DSNs.
dsn	The name of the DSN to be converted.
/suffix	An optional switch that changes the way DSNs are named. If this switch is used, the original DSN is retained and the new DSN is named "<*dsn*>-<*suffix*>."
suffix	The suffix that is used to name the new DSN.

Bookmark and bulk support for ODBC and OLE DB

Sybase supports bookmarks and SQL bulk operations for the ODBC Driver and the OLE DB Provider.

For the ODBC Driver

Sybase supports bulk insertions that use SQLBulkOperations with the option of SQL_ADD and cursor positioned updates & deletes using SQLSetPos (SQL_UPDATE, SQL_DELETE, SQL_POSITION). For instructions on using SQL_ADD and SQLSetPos, refer to the Microsoft Developer Network library's ODBC *Programmer's Reference*:

- SQL_ADD at http://msdn2.microsoft.com/en-us/library/ms712550.aspx
- SQLSetPos at http://msdn2.microsoft.com/en-us/library/ms713507.aspx

For the OLE DB Provider

Sybase supports bookmark operations that use the IRowsetLocate interface, which provides methods for comparing bookmarks and retrieving rows, based on bookmarks. For instructions on using IRowsetLocate, refer to the MSDN OLE DB Programmer's Reference at http://msdn2.microsoft.com/en-us/library/ms721190.aspx.

Sybase *interfaces* and *sql.ini* file support for jConnect

This new feature supports the use of the *sql.ini* file (for Windows) and the *interfaces* file (for UNIX) to provide server information for jConnect for JDBC.

Currently, to connect to the driver or provider you must specify several properties, such as server name or IP address and port number of an ASE server. By using the *sql.ini* or *interfaces* file, enterprises can centralize the information about the services available in the enterprise networks including ASE server information.

Modifying the connection string

This section describes what you must add to the connection string to identify the *sql.ini* or *interfaces* file. On jConnect for JDBC, you may only connect to a single Directory Services URL (DSURL).

Note jConnect does not support multiple-URLs.

Connection string for a single DSURL for jConnect

For a single DSURL for jConnect, add the following properties to the connection string in the following format:

Note You must specify the path to the *sql.ini* file and the server name.

```
String url = "jdbc:sybase:jndi:file://D:/syb1252/ini/mysql.ini?myaseISO1”
```

where:

- server name = *myaseISO1*.
- *sql.ini* file path = *file://D:/syb1252/ini/sql.ini*.

If the *sql.ini* path or server name is not specified in the URL, the driver returns an error.

Format for the interfaces file for SSL

The following is the format for the *sql.ini* file for SSL:

```
[SYBSRV2]
master=nlwnsck,mango1,4100,ssl
query=nlwnsck,mango1,4100,ssl
```

```
query=nlwnsck,mango1,5000,ssl
```

Note jConnect supports multiple query entries under the same server name in the *sql.ini* file. jConnect attempts to connect to values for host or port from the query entry in the sequence, as in the *sql.ini* file. If jConnect finds an SSL in a query entry, it will require the application to be coded to handle SSL connections by specifying an application specific socket factory, or the connection may fail.

Open Server 15.0 and SDK 15.0 features

This section describes the features that are available for the 15.0 release.

Open Client and Open Server 15.0 features

This section describes the features that are available for Open Client and Open Server in the 15.0 release:

Sybase library names changed

Naming conventions for Open Server and SDK libraries have changed to include "*syb*" in library names. This differentiates Sybase libraries from other frequently-used libraries, such as *libtcl.so* libraries.

To illustrate, instead of library names beginning with "*lib*" (for example, *libsrv.so*), library names now begin with "*libsyb*" (for example, *libsybsrv.so*). This uniquely identifies the Sybase libraries.

Sybase provides the scripts to facilitate migration to the new naming convention. The scripts are *lnsyblibs* for UNIX and *copylibs.bat* for Microsoft Windows:

- For UNIX, the *lnsyblibs* script creates softlinks between old and new library names, allowing pre-15.0 applications to work with renamed libraries.
- For Windows, the copylibs.bat script copies the required **.dll* files, allowing the renamed libraries to be used.

lnsyblibs

Script location:

$SYBASE/$SYBASE_OCS/(dev)lib

Script usage:

lnsyblibs { **create** | **remove** }

For the 15.0 release, create makes the symbolic links in *$SYBASE/$SYBASE_OCS/(dev)lib*.

copylibs.bat

Script location:

%SYBASE%\%SYBASE_OCS%\dll

Script usage:

copylibs.bat { **create** | **remove** }

For the 15.0 release:

- create copies the old-named files in *$SYBASE/$SYBASE_OCS/dll*, and
- remove can be used to delete these files.

Open Server and SDK libraries affected

The library name change affects the following components: DB-Library (*dblib*), Client-Library (*ctlib*), ESQL/C, ESQL/COBOL, Open Server, and XA.

Directory structure for this version

All Open Server and Open Client 15.0 products are now available in the following directory locations:

- UNIX: *$SYBASE/$SYBASE_OCS*
- Windows: *%SYBASE%\%SYBASE_OCS%*

For the 15.0 release, *$SYBASE_OCS* (or *%SYBASE%* on Windows) is *OCS-15_0*.

For more information, see the SDK and Open Server 15.0 *Installation Guide* for your platform.

Socket/tli drivers embedded in network library

In previous releases, dynamically-loaded network driver libraries, *libtli.so* or *libibsck.so*, were available and had to be used. In 15.0, these libraries are not available as separate driver libraries. Instead, they are embedded in the network library (*libsybtcl.so* or *libsybtcl.a*).

Unilib supported

Sybase now includes Unicode Infrastructure Library (Unilib®), an independent library of Unicode-based routines, to facilitate character set conversion.

For this release, Sybase ships a *libsybunic* library. To use it, you must re-link all applications with the new library unless you link with shared libraries.

For more information on Unilib support, see the Open Client and Open Server 15.0 *International Developer's Guide*.

New datatypes supported

The following sections describe the new datatypes that are supported by Open Client 15.0 and Open Server for 15.0.

For more information on the new datatypes, see these documents:

- Open Client 15.0 *Embedded SQL/C Programmer's Guide*
- Open Client 15.0 *Embedded SQL/COBOL Programmer's Guide*
- Open Client 15.0 *Client-Library/C Programmer's Guide*
- Open Client 15.0 *Client-Library/C Reference Manual*
- Open Server 15.0 *Server-Library/C Reference Manual*

Bigint datatype

A new C-programming, integer datatype named CS_BIGINT is included. CS_BIGINT is a signed, 8-byte integer datatype for use on 32-bit UNIX platforms.

Unsigned *int* datatypes

The following new C-programming, unsigned integer datatypes are included:

- CS_USMALLINT (2-byte unsigned integer)
- CS_UINT (4-byte unsigned integer)
- CS_UBIGINT (8-byte unsigned integer)

Sybase provides related conversion routines to access these unsigned int datatypes from the data server.

Unitext datatype

A new C-programming server-defined datatype named CS_UNITEXT is included.

CS_UNITEXT is an unsigned, short datatype that contains related conversion routines to access the 2-byte Unicode UTF-16 data from the server. This datatype follows the same rules as CS_TEXT, but in a fixed UTF-16 encoding, regardless of the Adaptive Server default character set.

The variable-length unitext datatype can hold up to 1,073,741,823 Unicode characters (2,147,483,646 bytes).

XML datatype

A new C-programming, server-defined datatype named CS_XML is included.

CS_XML is an internal C-language, unsigned character datatype for all platforms. It includes related conversion routines to access the variable-width XML data from the server.

The behavior of this datatype is similar to the standard CS_TEXT and CS_IMAGE datatypes, except it represents XML data.

Syntactically, CS_XML can be used anywhere that CS_TEXT and CS_IMAGE are used.

Note ASE 15.0 does not currently support the XML datatype.

Server packet size increased

You can now configure the Adaptive Server and Open Server packet size. A larger packet size causes fewer I/O operations, potentially improving performance. Also, the Open Server default has increased to 8192 bytes.

BCP partition support for multiple-partition, multiple-file operations

BCP supports transfer of data to and from partitions of tables in ASE. Data can also be copied to and from multiple data files in parallel.

The following section describes changes to BLK_PARTITION for BCP partition support.

BLK_PARTITION changes

To support named BCP partitions in BCP_IN and BCP_OUT operations, Sybase added a new property, BLK_PARTITION. The syntax for blk_props with the new BLK_PARTITION property follows:

```
blk_props(blkdescr, CS_SET, BLK_PARTITION, partbuf,
CS_NULLTERM, NULL);
```

You must consider the following issues when using BLK_PARTITION:

- The partbuf buffer must be a pointer to a character string containing the name of the partition.
- Only one name may be provided. A single BLKLIB operation always operates on an entire table or on a single partition. If no partition name is provided, BLKLIB will not operate on a specific partition but on the entire table.
- This property can be used for both BCP_IN and BCP_OUT operations, whereas the BLK_SLICENUM property can only be used for BCP_IN. Either BLK_PARTITION or BLK_SLICENUM can be used; if one is set, the other is cleared.
- The BLK_PARTITION property does not require you to set CS_VERSION_150 or BLK_VERSION_150.
- A new option, partition *partition_name*, is added to the bcp syntax. The option is the name of the partition in Adaptive Server.

bcp syntax changes

For the latest bcp syntax, see Appendix A, "Utility Commands Reference," in the Open Client and Open Server 15.0 *Programmer's Supplement* for your platform.

For more information on bcp partition support, see these documents:

- Open Client and Open Server 15.0 *Programmer's Supplement* for your platform
- Open Client and Open Server 15.0 *Common Libraries Reference Manual*

BCP support for computed columns

BCP provides support for ASE tables with the following computed columns:

- Materialized Computed Columns (MCC): BCP treats these as ordinary columns and transfers data in and out of ASE.
- Virtual Computed Columns (VCC): You can instruct BCP to not include VCC data while transferring data in and out of ASE.

- Functional Index (FI): You can instruct BCP to include FI data while transferring data in and out of ASE.

Client-Library changes

Two new Client-Library options are included to support BCP computed columns: CS_OPT_SHOW_FI exposes the columns of each FI, and CS_OPT_HIDE_VCC instructs the server to hide VCCs.

bcp syntax changes

Two new parameters are included to support BCP computed columns:

- --hide-vcc: Instructs BCP not to copy VCCs.
- --show-fi: Instructs BCP to copy FIs.

For more information on BCP computed columns, see these documents:

- Open Client and Open Server 15.0 *Programmer's Supplement* for your platform
- Open Client and Open Server 15.0 *Common Libraries Reference Manual*
- Open Client 15.0 *Client-Library/C Reference Manual*

Dynamic version of bulk library available

Sybase now ships a dynamic version of bulk library for UNIX platforms. Bulk library is no longer directly dependent on Open Server libraries. Both the reentrant (*libsybblk_r.so*) and non-reentrant (*libsybblk.so*) libraries are available as shared libraries.

Large identifiers supported

Identifiers are database object names, such as server names, columns names, and table names. Limits on lengths of identifiers have been increased to 255 bytes. For example, the length of CS_MAX_NAME is now 255 bytes. However, limits on login names and passwords remain at 30 bytes.

For more information on large identifiers, see these documents:

- Open Client 15.0 *Client-Library/C Programmer's Guide*
- Open Client 15.0 *Client-Library/C Reference Manual*
- Open Server 15.0 *Server-Library/C Reference Manual*

IPv6 supported

Sybase now supports IPv6 on the following platforms:

- HP-UX 11.11 32-bit and 64-bit
- IBM AIX 5.2 32-bit and 64-bit
- Linux x86 32-bit
- Sun Solaris 8 32-bit and 64-bit
- Windows x86 32-bit (2003)

The supported platforms are for the following products:

- SDK components:
  - Open Client (Client-Library)
  - ESQL/C
  - ESQL/COBOL
- Open Server

The LDAP directory service driver does not support the use of IPv6 to connect to an LDAP server.

Client-Library scrollable cursors

The scrollable cursor feature provides a way to set the current position anywhere in the result set by specifying a NEXT, PREVIOUS, FIRST, LAST, ABSOLUTE or RELATIVE clause in a FETCH statement. It implements a scrollable cursor that is read-only with either an INSENSITIVE or a SEMI_SENSITIVE property.

Open Client allows both scrollable and non-scrollable cursors. Scrollable cursors can be insensitive or semi_sensitive. Non-scrollable cursors in 15.0 can have an insensitive attribute. Pre-15.0 cursors have no sensitivity attributes.

Implementation uses either the old-style ct_fetch() API for non-scrollable cursors or insensitive non-scrollable cursors. A new function named ct_scroll_fetch() has been added to the API to cater to scrollable cursors. Both API allow for single or multi-row fetches.

Configuration options

No configuration options are required. During client login, a capability bit is tested to see if the connecting ASE supports scrollable cursors. If no support is available, the ct_scroll_fetch() API cannot be used; instead, the application must use ct_fetch() instead.

For more information on Client-Library scrollable cursors, see these documents:

- Open Client 15.0 *Client-Library/C Reference Manual*
- Open Client 15.0 *Client-Library/C Programmer's Guide*

Supported COBOL compiler version changed

The supported COBOL compiler is changed to MicroFocus COBOL 4.0 on UNIX platforms and to Net Express 4.0 on Microsoft Windows.

SSL Plus 5.0.4 + SBGSE 2.0

Certicom Secure Sockets Layer (SSL) Plus 5.0.4 + SBGSE 2.0 support is provided for the following Sybase products:

- Open Server
- SDK components:
  - Open Client (Client-Library)
  - ESQL/C
  - ESQL/COBOL

Support for SSL Plus 5.0.4 is available on the following platforms:

- IBM RS/6000 AIX 5.2 32-bit and 64-bit
- HP-UX 11.11 32-bit and 64-bit
- Linux x86 32-bit
- Sun Solaris 8 (SPARC) 32-bit and 64-bit
- Windows x86 32-bit (2000 Service Pack 4 or later)
- Windows x86 32-bit (2003 Service Pack 1 or later)
- Windows x86 32-bit (XP Service Pack 2 or later)

SBGSE 2.0 is supported on all SSL Plus 5.0.4 platforms, with the exception of Linux x86 32-bit.

As a result of U.S. Federal regulations, Sybase is required to use FIPS-certified cipher suites. After you upgrade to SDK 15.0 and to Open Server 15.0, perform the following:

- For Sun Solaris 8 32-bit and 64-bit, HP-UX 11.11 32-bit and 64-bit, and IBM AIX 5.2 32-bit and 64-bit, be sure that the *lib3p* or *lib3p64* directory is in the dynamic load library path.

- In Windows, to allow the *ctlib* and *srvlib* libraries to find the Certicom cipher suites in *sbgse2.dll*, add the following to the dynamic load library path:

  ```
  %SYBASE%\%SYBASE_OCS%\lib3p
  ```

  When running any application that uses SSL, be sure that the *lib3p* directory is in its dynamic load library path.

DB-Library changes

A new DB-Library routine, dbsetconnect, allows you to specify server connection information in this routine. This routine, when used in a client application, ignores the server connection information specified in any of these places:

- As an interfaces entry that corresponds to the value of the DSQUERY environment variable.
- As directory services lookup for a server entry in the interfaces file on UNIX platforms, or the LDAP directory services, and the registry file on Microsoft Windows.

A new DB-Library environment variable, SYBOCS_DBVERSION, allows you to externally configure the DB-Library version level at runtime.

DB-Library uses the new environment variable to perform the following:

- Retrieve the environment variable at the DB-Library initialization stage
- Store the environment variable value at the version level

To configure SYBOCS_DBVERSION, see the Open Client and Open Server 15.0 *Configuration Guide* for your platform.

New environment variable

A new environment variable, SYBOCS_CFG, allows you to override the following default external configuration file paths:

- UNIX: *$SYBASE/$SYBASE_OCS/config/ocs.cfg*
- Windows: *%SYBASE%\%SYBASE_OCS%\ini\ocs.cfg*

To configure SYBOCS_CFG, see the Open Client and Open Server 15.0 *Configuration Guide* for your platform.

SDK 15.0 features for the ASE Drivers and Data Providers

This section describes the new features that you can use for the following components in the 15.0 release:

- ASE ODBC Driver by Sybase for Windows and Linux
- ASE ADO.NET Data Provider by Sybase for Windows
- ASE OLE DB Data Provider by Sybase for Windows

The following new features are supported in this version:

- New datatypes
- Computed columns
- Large identifiers
- Server-side scrollable cursors
- ASE default network packet size
- Default client character set (Linux only)
- SSL Plus 5.0.4 + SBGSE 2.0 support on Windows

For more information on the new features, see these documents:

- Adaptive Server Enterprise ODBC Driver by Sybase *User's Guide* for your platform

Note The ODBC Driver Kit included with previous versions of SDK is no longer shipped. Instead, for SDK 15.0, the ODBC Driver Kit is superseded with the ASE ODBC Driver by Sybase.

The ASE ODBC Driver by Sybase was first introduced with SDK 12.5.1, and shipped with the ODBC Driver Kit. The ODBC Driver Kit was installed in *%SYBASE%\ODBC*, and is registered with the ODBC Driver Manager as "Sybase ASE ODBC Driver".

The ASE ODBC Driver by Sybase is installed in *%SYBASE%\DataAccess\ODBC*, and is registered as "Adaptive Server Enterprise". The version shipping with SDK 15.0 is 15.0.0.50.

- Adaptive Server Enterprise OLE DB Provider by Sybase *User's Guide* for Microsoft Windows

Note The OLE DB Driver Kit included with previous versions of SDK is no longer shipped. Instead, for SDK 15.0, OLE DB Driver Kit is superseded with the ASE OLE DB Provider by Sybase.

The ASE OLE DB Provider by Sybase was first introduced with SDK 12.5.1, and shipped with the OLE DB Driver Kit. The OLE DB Driver Kit was installed in *%SYBASE%\OLEDB*, and used the provider short name of "Sybase.ASEOLEDBProvider" and the long name of "Sybase ASE OLE DB Provider."

The ASE OLE DB Provider by Sybase is installed in *%SYBASE%\DataAccess\OLEDB*, and uses provider short name "ASEOLEDB." The version shipping with SDK 15.0 is 15.0.0.51.

- Adaptive Server Enterprise ADO.NET Data Provider *User's Guide*

Migrating to ASE ODBC Driver by Sybase and ASE OLE DB Provider by Sybase

This section describes how to migrate to the ASE ODBC Driver by Sybase and ASE OLE DB Provider by Sybase.

Migrating to ASE ODBC Driver by Sybase

To migrate your applications from the ODBC Driver Kit to the ASE ODBC Driver by Sybase, be sure to complete the following steps:

1 Migrate DSNs.

You must recreate the DSNs you are using to utilize the ASE ODBC Driver by Sybase. Alternatively, you can create new DSNs with different names and change the DSN names in the application code.

2 Migrate ODBC application code.

If you created new DSNs with different names, you must change the DSN names in the SQLConnect calls. You must also change the SQLDriverConnect connection string if you use `Driver=`***`Driver Name`*** as the driver name. The name of the ASE ODBC Driver by Sybase is "Adaptive Server Enterprise."

The ODBC DSN Migration tool helps you to migrate from the ODBC Driver Kit to the ODBC Driver by Sybase. More information on the tool is available in "ODBC DSN Migration tool available" on page 133.

Known differences in behavior between the ODBC Driver Kit and the ASE ODBC Driver by Sybase are documented in the SDK Version 15.0 *Release Bulletin* for your platform.

Note The connection string syntax for ASE ODBC Driver by Sybase is documented in the Adaptive Server Enterprise ODBC Driver by Sybase *User's Guide* for your platform.

The connection string syntax differs from the syntax for the ODBC Driver Kit. The ODBC Driver by Sybase honors the ODBC Driver Kit syntax, but Sybase recommends that you migrate your connection string syntax to the new syntax when possible.

Migrating to ASE OLE DB Provider by Sybase

To migrate OLE DB applications to use the ASE OLE DB Provider by Sybase, you must edit the connection string used by OLE DB client applications. The provider short name for the ASE OLE DB Provider by Sybase is "ASEOLEDB."

Known differences in behavior between the OLE DB Driver Kit and ASE OLE DB Provider by Sybase are documented in the SDK 15.0 *Release Bulletin* for your platform.

Note The connection string syntax for ASE OLE DB Provider by Sybase is documented in the Adaptive Server Enterprise OLE DB Provider by Sybase *User's Guide* for Microsoft Windows.

The connection string syntax differs from the syntax for the OLE DB Driver Kit. The OLE DB Provider by Sybase honors the OLE DB Driver Kit syntax, but Sybase recommends that you migrate your connection string syntax to the new syntax when possible.

ADO.NET Data Provider directory change and migration

ASE ADO.NET Data Provider 1.15 shipped with this SDK release is installed in the *%SYBASE%\DataAccess\ADONET* directory. This has changed from the previous installation location, where it was installed in *%SYBASE%\ADO.NET*.

To migrate your .NET applications to use the new data provider, you must rebuild your application using ASE ADO.NET Data Provider 1.15.

New datatypes

ASE supports three new datatypes:

- bigint – an exact numeric datatype designed to be used when the range of the existing int types is insufficient.
- unsigned int – unsigned versions of the following signed exact numeric integer datatypes: unsignedsmallint, unsignedint, and unsignedbigint.
- unitext – a variable-length datatype for Unicode characters.

The following table describes all new datatypes supported by ASE, SDK Data Providers, and Drivers:

ASE datatype	Stores data which is	Size in bytes
bigint	Whole numbers between -2^{63} and $2^{63} - 1$ (-9,223,372,036,854,775,808 and +9,223,372,036,854,775,807)	8
unsignedsmallint	0 and 65535	2
unsignedint	0 and 4,294,967,295	4
unsignedbigint	0 and 18,446,744,073,709,551,61)	8
unitext	Unicode characters up to 30^{-1} characters in length (1,073,741,823)	231^{-1} (2,147,483,646)

For more information on the new ASE datatypes, see these documents:

- Adaptive Server Enterprise ODBC Driver by Sybase *User's Guide* for your platform
- Adaptive Server Enterprise OLE DB Provider by Sybase *User's Guide* for Microsoft Windows
- Adaptive Server Enterprise ADO.NET Data Provider *User's Guide*

Computed columns supported

The ASE Drivers support computed columns that allow you to create a shorthand term for an expression, such as "Pay" for "Salary + Commission," and to make that column indexable, as long as its datatype can be indexed. Computed columns are defined by an expression, whether from regular columns in the same row, functions, arithmetic operators, and path names, including their metadata information.

Large identifiers supported

The ASE Drivers support the new ASE large identifiers, or names, for database objects. Some object names in ASE 15.0 now have new limits of 255 bytes. For example, you can now have longer names for tables, columns, procedures, and others.

If you use large identifiers in C++ programs or client applications, you must allocate sufficient buffer lengths to avoid data truncation.

Server-side scrollable cursors supported

Scrollable cursors allow applications to set the current position of the cursor anywhere in the result set by specifying appropriate scrolling options. Applications can use one of NEXT, PRIOR, FIRST, LAST, RELATIVE, or ABSOLUTE scrolling options to traverse the result set as desired.

Beginning with this release, Adaptive Server provides native support for static insensitive scrollable cursors on the server.

In this release, ASE ODBC and OLE DB Drivers can also use these server-side scrollable cursors. Server-side scrollable cursors are more efficient in the use of network and memory resources as compared to client-side scrollable cursors.

Server-side scrollable cursors can be used by the ASE ODBC and OLE DB Drivers when the following conditions are met:

- You are connected to Adaptive Server 15.0 or later.
- The UseCursor connection property is set to 1.

ASE default network packet size increased

The network packet size used by ASE Drivers to communicate with Adaptive Server is now set based on the default network packet size provided by Adaptive Server. This is negotiated as part of the login process.

This new process allows you to configure the network packet size used by applications centrally on Adaptive Server.

Increased packet size requires less I/O and potentially improves performance.

If you have memory restrictions on the client, you can set the RestrictMaximumPacketSize property to the maximum memory allowable for network buffers. In previous versions, drivers would default to 512 bytes as the packet size.

Default client character set changed (Linux only)

By default, the ASE ODBC driver on Linux now uses the charset configured on the client machine.

You can override this by specifying `CharSet=`***`ServerDefault`***, in which case the driver uses the ASE default charset. Alternatively, you can set the `CharSet` property to any ASE-supported charset.

On Windows, the driver uses the default charset configured on the server.

SSL Plus 5.0.4 + SBGSE 2.0 support on Windows

In this release, ASE Drivers on the Windows platform use Secure Sockets Layer (SSL) Plus 5.0.4 + SBGSE 2.0 for SSL connections. As a result, FIPS certified cipher suites are used in SSL connections.

SDK 15.0 features for jConnect

This SDK release includes jConnect 6.05, which supports the following ASE 15.0 features:

- New datatypes
- Computed columns
- Large identifiers
- Scrollable cursors
- ASE default network packet size
- Directory services *sql.ini* interface file support

In addition, jConnect 6.05 supports the following JDBC 3.0 features:

- Parameter MetaData
- Named Parameters

This section describes the ASE 15.0 features supported by jConnect 6.05.

For more information on these features, see version 6.05 or higher of the jConnect for JDBC *Programmer's Reference*.

New datatypes supported

jConnect supports the following new datatypes:

- bigint – an exact numeric datatype designed to be used when the range of the existing int types is insufficient.
- unsigned int – unsigned versions of the exact numeric integer datatypes: unsignedsmallint, unsignedint, and unsignedbigint.
- unitext – a variable-length datatype for Unicode characters.

Bigint datatype

Sybase supports bigint, which is a 64-bit integer datatype that is supported as a native ASE datatype. In Java, this datatype maps to Java datatype long. To use this as a parameter, you can call PreparedStatement.setLong(int index, long value) and jConnect sends the data as bigint to ASE. When retrieving from a bigint column, you can use the ResutSet.getLong(int index) method.

Unitext datatypes

There are no API changes in jConnect for using the unitext datatype. jConnect can internally handle storage and retrieval of data from ASE when unitext columns are used.

Unsigned int datatypes

In this release, ASE has introduced unsigned bigint, unsigned int, and unsigned smallint as native ASE datatypes. Because there are no corresponding unsigned datatypes in Java, you must set and get the next higher integer if you want to process the data correctly.

For example, if you are retrieving data from an unsigned int, using the Java datatype *int* is too small to contain positive large values, and as a result, ResultSet.getInt (int index) might return incorrect data or throw an exception. To process the data correctly, you should get the next higher integer value ResultSet.getLong().

You can use the following table to set or get data.

ASE datatype	Java datatype
unsigned smallint	setInt(), getInt()
unsigned int	setLong(), getLong()
unsigned bigint	setBigDecimal(), getBigDecimal()

Computed columns supported

Computed columns allow you to create a shorthand term for an expression, such as “Pay” for “Salary * Commission,” and to make that column indexable, as long as its datatype can be indexed. Computed columns are defined by an expression, whether from regular columns in the same row, or from functions, arithmetic operators, path names, and others.

The jConnect Driver supports accessing these specific computed columns, including their metadata information.

Large identifiers supported

The new limit for the length of object names or identifiers is 255 bytes. The new large identifier applies to most user-defined identifiers, including table name, column name, index name, and so on. As a result of the expanded limits, Sybase has expanded some system tables (catalogs) and built-in functions.

The jConnect driver fully supports these large identifiers in your applications.

Scrollable cursors supported

Applications written using jConnect use the scrollable cursor functionality with the JDBC ResultSet, Statement, and Connection APIs. These APIs are already supported in jConnect, but because ASE did not natively support scrollable cursor functionality, it was implemented using client-side row set caching. With this release, jConnect determines if the ASE connection supports native scrollable cursor functionality and uses it instead of client-side caching. As a result, most applications can expect significant performance gain in accessing out-of-order rows and reduction in client-side memory requirements.

Note jConnect will continue to use client-side caching for older Adaptive Server releases.

This release supports the following scrollable cursors:

- Read-only (CONCUR_READ_ONLY)
- Insensitive (TYPE_SCROLL_INSENSITIVE)

ASE default network packet size increased

The network packet size used by jConnect to communicate with Adaptive Server is now based on the default network packet size provided by the Adaptive Server. This is negotiated as part of the login process. In previous versions, the drivers would default to 512 bytes as the packet size.

This new process allows you to configure the network packet size used by applications centrally in the Adaptive Server. Increased packet size requires less I/O and potentially improves performance. If you have memory restrictions on the client, you can set the PACKETSIZE connection property to the maximum memory allowable for network buffers.

Accessibility features

Section 508 requires that U.S. Federal agencies' electronic and information technology is accessible to people with disabilities. Sybase strongly supports Section 508 and has made a range of Sybase products Section 508-compliant, including Open Client and Open Server version 15.0.

Documents in the 15.0 release are available in HTML 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.

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.