Release Bulletin
Open Client/Server™ Products Version 11.1.1
for Linux/Intel

Table of Contents

1. Accessing Current Release Bulletin Information 1
2. Product Summary 2
3. Special Installation Instructions 3
4. Known Problems 7
5. Product Compatibilities 7
6. Documentation Updates and Clarifications 9
7. Programming Issues 18
8. Utility Issues 19
9. Technical Support 20
10. Other Sources of Information 20

1. Accessing Current Release Bulletin Information

A more recent version of this release bulletin may be available on the World Wide Web. To check for critical product or document information added after the release of the product CD, use the Sybase Technical Library Product Manual Web site.

To access release bulletins at the Technical Library Product Manual Web site:
1. Go to support.sybase.com.
2. Under the Support Services heading, choose Product Manuals.
3. Choose the appropriate version link.
4. In the window on the right, choose the “platform” collection of the product you are interested in.
5. In the window on the right, choose the release bulletin for your platform.

2. Product Summary

Enclosed is Open Client/Server version 11.1.1, which is compatible with the following platform and operating system configuration:
• Linux Intel kernel installation 2.2.5
• glibc run-time environment 2.0.7-29

2.1. Version Contents

• Open Client™/C version 11.1.1
• Open Server™ version 11.1.1

2.2. Drivers

The names of some drivers listed in the Open Client/Server Configuration Guide for UNIX have changed, and some new drivers are provided. Table 1 lists the Sybase® drivers included with Open Client/Server version 11.1.1, and compatibilities for third-party services.

<table>
<thead>
<tr>
<th>Driver</th>
<th>Name</th>
<th>Service Compatibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Independent Protocol Driver</td>
<td>libinsck.so</td>
<td>TCP/IP Sockets</td>
</tr>
</tbody>
</table>

If you want to use Open Client/Server version 11.1.1 directory service support, you must use the appropriate software to support that service.
3. Special Installation Instructions

This section describes installation and configuration issues. See the Open Client/Server Configuration Guide for UNIX for instructions on initializing the software and configuring the interfaces file.

For additional information on configuring Open Client/Server products, see the Open Client/Server Programmer’s Supplement for UNIX.

➤ Note

Linux-specific issues are not addressed in these UNIX documents, but are instead discussed in this release bulletin.

3.1. Installation and Configuration Instructions

Follow these instructions to install the Open Client/Server software. When the software has loaded successfully (the installation has completed with no errors), the product is ready to use.

➤ Note

Your Customer Authorization String (CAS) is part of the Red Hat Package Manager (RPM) installation. You do not need to manually enter this string at the beginning of the installation process.

3.1.1. Determine the Size and Location of the Sybase Installation Directory

The installation will target the /opt directory by default. Verify that you have enough free file-system space on your device before you install the new software.

Table 2: Sybase Installation Directory Space Requirements

<table>
<thead>
<tr>
<th>Product</th>
<th>Linux/Intel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open Client/C</td>
<td>42 MB</td>
</tr>
<tr>
<td>Open Server</td>
<td>14 MB</td>
</tr>
<tr>
<td>Sybase Enterprise Software Common Files</td>
<td>14 MB</td>
</tr>
</tbody>
</table>
3.1.2. rsh Must Work on Install Host

You must be able to rsh into the machine on which you’re installing the software. If you can’t do this, add the install host to your .rhosts file.

3.1.3. Checking for Shared Libraries

Check that the following shared libraries are present on your machine:

- libnsl.so.1
- libm.so.6
- libdl.so.2
- libc.so.6
- /lib/ld-linux.so.2

The X run-time system, normally included in the Linux operating system distribution, must be present before sybsetup can be used.

If the libraries are not in their default locations, make sure that the loader can find them. You can place a symbolic link in the default location that points to the actual location of the libraries. Or, you can set the LD_LIBRARY_PATH environment variable to the directory or directories where the libraries are stored. For example, enter this command at the UNIX C-shell prompt:

```
setenv LD_LIBRARY_PATH library_path1:library_path2
```

where library_path is the path to a shared library directory.

3.1.4. Creating a Sybase DBA and SA Account

Before you install any Sybase product, you must create a Sybase user and group account. Follow these steps:

1. Log in as “root”.
2. Enter:
# groupadd sybase
# useradd -g sybase -c "Sybase DBA account" sybase

3.1.5. Setting the CD format

The CD format is ISO 9660 compliant, and your system reads this format by default. If you experience CD reading errors, check your kernel to make sure the ISO 9660 option is turned on.

3.2. Installing with RPM

In the Linux environment, you can unload Open Client/Server with either the Red Hat Package Manager (RPM) or Sybase’s sybsetup utility.

Follow the instructions in this section to unload Open Client/Server products using the RPM, which is bundled with Linux by all distributors.

➤ Note

The Sybase Enterprise Software related products install, by default, under /opt/sybase-11.9.2, as indicated by the product’s internal version strings. To install the product somewhere else, use the relocate option. For example:

```bash
# rpm -hiv --relocate 
/opt/sybase-11.9.2=/usr/local/sybase 
/mnt/cdrom/RPMS/sybase-common-11.9.2-1.i386.rpm
```

1. Verify that you are logged in as “root” and that you have superuser privileges.
2. If you decide to install under the default /opt filesystem or directory, verify that you have a directory called /opt on your server, and that it has sufficient space to accommodate unloading the software. If an /opt directory does not already exist, the Linux RPM will create it in the root directory.
3. Place the Open Client/Server product CD in the CD-ROM drive.
4. At the # prompt, enter:

```bash
# mount -t iso9660 /dev/cdrom /mnt/cdrom
```
5. To get initial help on RPM, enter:

```bash
# /bin/rpm
```
6. Unload all of your Open Client products into the default 
/opt/sybase-11.9.2 directory by entering:

```
# rpm -hiv /mnt/cdrom/RPMS/sybase-openclient-11.1.1-1.i386.rpm
```

7. Unload all of your Open Server products into the 
/opt/sybase-11.9.2 directory by entering:

```
# rpm -hiv /mnt/cdrom/RPMS/sybase-openserver-11.1.1-1.i386.rpm
```

8. When unloading is complete, unmount the CD. Enter:

```
# umount /mnt/cdrom
Remove the CD. Enter:

# eject
```

**Installing the Open Client Run-Time Environment**

All Sybase Enterprise products require the Sybase Open Client run-time environment. This environment is available in your sybase-common-11.1.1-1.i386.rpm package. If you need only this run-time environment, you do not need to install Open Client or Open Server.

Before Sybase Open Client or Sybase Open Server can be installed, you must install the Sybase Common package. It provides the Open Client runtime environment on which all other packages depend.

3.2.1. Starting sybsetup from CD-ROM

When you start sybsetup, it searches for the DISPLAY environment variable and window manager. If sybsetup cannot locate those items, you can continue in character mode, or you can exit and reset the necessary environment variables.

To start sybsetup:

1. Place the Sybase products CD in the CD-ROM drive.
2. Log in to the machine to which a CD-ROM drive is physically connected, using “root” superuser as your login ID.
3. Mount the CD. Enter:
   
   ```
   # mount -t iso9660 /dev/cdrom /mnt/cdrom
   ```
4. Follow the rest of the installation instructions in the section titled “Starting sybsetup from CD-ROM or Tape” in Chapter 2 of Installing Open Client/Server Products on UNIX Platforms.
5. Use this command to start sybsetup:
3.3. Modifying interfaces File Entries with the dscp Utility

Use dscp to modify master and query server entries in the interfaces file.

4. Known Problems

For complete listings of known problems and fixed bugs, see the following files in the $SYBASE/install/SPR directory:

Table 3: File names for known problems

<table>
<thead>
<tr>
<th>Product</th>
<th>File Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client-Library™</td>
<td>spr_ctlib</td>
<td>System problem reports for Client-Library</td>
</tr>
<tr>
<td></td>
<td>cpr_ctlib</td>
<td>Closed problem reports for Client-Library</td>
</tr>
<tr>
<td>DB-Library™</td>
<td>spr_dblib</td>
<td>System problem reports for DB-Library</td>
</tr>
<tr>
<td></td>
<td>cpr_dblib</td>
<td>Closed problem reports for DB-Library</td>
</tr>
<tr>
<td>Open Server</td>
<td>spr_srvlib</td>
<td>System problem reports for Open Server-Library</td>
</tr>
<tr>
<td></td>
<td>cpr_srvlib</td>
<td>Closed problem reports for Open Server-Library</td>
</tr>
<tr>
<td>CS-Library™</td>
<td>spr_cslib</td>
<td>System problem reports for CS-Library</td>
</tr>
<tr>
<td></td>
<td>cpr_cslib</td>
<td>Closed problem reports for CS-Library</td>
</tr>
</tbody>
</table>

5. Product Compatibilities

This section provides information about compatibilities among Sybase products.
5.1. Client-Library Compatibility

Client-Library version 11.1.1 is certified to work with the Open Server, Sybase Adaptive Server™, and SQL Server™ products shown in Table 4.

Table 4: Client-Library compatibility

<table>
<thead>
<tr>
<th>Platform</th>
<th>Open Server 11.1.1</th>
<th>Open Server 10.0.4</th>
<th>Adaptive Server 11.5</th>
<th>SQL Server 11.0.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital UNIX 4.0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>HP 9000/800 HP-UX 10.0.1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>HP 9000/800 HP-UX 11.0</td>
<td>X</td>
<td>n/a</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>IBM RS/6000 AIX 4.1.4</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Linux/Intel</td>
<td>X</td>
<td>X</td>
<td>11.9.2</td>
<td>X</td>
</tr>
<tr>
<td>Solaris 2.5.1 (SPARC)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Windows NT 3.51/4.0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Windows 95</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>Windows 3.11</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>

LEGEND for Table 4: X = compatible; blank = compatibility not tested; n/a = product not available on that platform.

In addition, note these compatibility issues for Open Client/C:

- Header files included in an application must be of the same version level as the library with which the application is linked.
5.2. Open Server Compatibility

Open Server version 11.1.1 is certified to work with the Open Client/C, Adaptive Server, and SQL Server products shown in Table 5.

Table 5: Open Server compatibility

<table>
<thead>
<tr>
<th>Platform</th>
<th>Client-Library 11.1</th>
<th>10.0.4</th>
<th>Adaptive Server 11.5</th>
<th>SQL Server 11.0.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital UNIX 4.0</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>HP 9000/800 HP-UX 10.0.1</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>HP 9000/800 HP-UX 11.0</td>
<td>X</td>
<td>n/a</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>IBM RS/6000 AIX 4.1.4</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Linux/Intel</td>
<td>X</td>
<td>X</td>
<td>11.9.2</td>
<td>X</td>
</tr>
<tr>
<td>Solaris 2.5.1 (SPARC)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Windows NT 3.51/4.0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Windows 95</td>
<td>n/a</td>
<td></td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>Windows 3.11</td>
<td>n/a</td>
<td></td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>

LEGEND for Table 5: X = compatible; blank = compatibility not tested; n/a = product not available on that platform.

In addition, note these compatibility issues for Open Server:
- Header files included in an application must be of the same version level as the library with which the application is linked.

6. Documentation Updates and Clarifications

This section contains updates and additions to the Open Client/Server documentation.

6.1. Installation Guide

In *Installing Open Client/Server Products on UNIX Platforms*, Table 1-3 lists Embedded SQL/C, Embedded SQL/COBOL and Security Guardian™. These are not available in the Linux release of Open Client/Server.
Table 2-1, *Installing Sybase Products*, step 2 says “In CD installations... *sybsetup* knows the CD device.” This is incorrect—you must type in the full path name to the CD device.

### 6.2. Bulk-Library: Bulk Copy Into Partitioned Tables

Bulk-Library is documented in the *Open Client and Open Server Common Libraries Reference Manual*. This information is not included in that manual.

Adaptive Server Enterprise supports partitioning of tables. For information on this feature, see the Adaptive Server *System Administration Guide* and *Performance and Tuning Guide*.

Bulk-Library includes new properties, BLK_IDSTARTNUM and BLK_SLICENUM to support bulk copy to partitioned tables. Also, the behavior of the BLK_IDENTITY property is affected by the addition of BLK_IDSTARTNUM. Table 6 describes these properties:

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th><em>buffer</em> Is</th>
<th>Applies To</th>
</tr>
</thead>
</table>
| BLK_IDENTITY     | Whether values for a table’s identity column are specified explicitly for each row to be inserted. This property cannot be set to CS_TRUE if BLK_IDSTARTNUM has been set for a bulk-copy-in operation. | CS_TRUE or CS_FALSE. The default is CS_FALSE, which indicates that identity values are either:  
  - Computed from the starting value indicated by BLK_IDSTARTNUM, or  
  - Computed by Adaptive Server as data is inserted, based on existing identity values in the table. | IN copies only |
Table 6: New and changed Bulk-Library properties (continued)

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>buffer</th>
<th>Applies To</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLK_IDSTARTNUM</td>
<td>The starting value for identity columns in inserted rows. The first inserted row uses this value, and the value is incremented for each subsequent row. This property cannot be set if BLK.IDENTITY has been set to CS_TRUE for the bulk-copy-in operation.</td>
<td>A CS_NUMERIC variable containing the starting identity value. There is no default.</td>
<td>IN copies only</td>
</tr>
<tr>
<td>BLK_SLICENUM</td>
<td>For bulk-copy into a partitioned table. Specifies the partition number that copied rows are inserted to. The default is CS_UNUSED, which indicates that Adaptive Server will randomly choose a partition number.</td>
<td>A CS_INT variable containing a positive value representing the partition number.</td>
<td>IN copies only</td>
</tr>
</tbody>
</table>

6.3. Signal Handling in Multithreaded Environments

This section provides information about signal handling for multithreaded applications on UNIX platforms. It supplements the Open Client/Server version 11.1 documentation that explains how to use Sybase libraries to build multithreaded applications.

6.3.1. Basic Concepts

UNIX operating systems use a signal to report an exceptional situation to a process. Some signals report synchronous events, such as references to an invalid address. Other signals report asynchronous events, such as the disconnection of a phone line.

You can specify an action to be taken when a signal occurs by installing a signal handler function. When the signal occurs, the operating system executes the signal handler function.

Use Sybase-provided calls to install signal handlers. If you use operating system calls to install signal handlers, it interferes with the internal workings of the Sybase libraries.
6.3.2. Signal Handling in Nonthreaded Environments

Signal handling is straightforward in a traditional, nonthreaded UNIX environment that uses version 10.0.4 or 11.1.1 nonthreaded Sybase libraries. Each process has a single thread of control. You register a handler for a given signal with Open Client/Server library calls. Use \texttt{ct\_callback()} in Client-Library and \texttt{srv\_signal()} in Server-Library.

When a signal occurs, the Sybase library traps the signal and calls the designated signal handler. To mask a signal, blocking it from delivery to a process, use the \texttt{sigprocmask()} UNIX system call.

6.3.3. Types of Signals

Signals fall into two categories that correspond to the events by which they are generated.

<table>
<thead>
<tr>
<th>Type of Event</th>
<th>Type of Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exception</td>
<td>Synchronous signal</td>
</tr>
<tr>
<td>External event</td>
<td>Asynchronous signal</td>
</tr>
</tbody>
</table>

Exceptions and Synchronous Signals

Synchronous signals are generated by exceptions, or errors, which are caused by invalid operations in a program. Examples of exceptions include attempts to access invalid memory addresses and attempts to divide by zero.

Examples of synchronous signals include SIGILL, SIGFPE, SIGBUS, SIGSEGV, SIGSYS, and SIGPIPE.

External Events and Asynchronous Signals

Asynchronous signals are generated by events outside the control of the process that receives them, and arrive at unpredictable times. Asynchronous signals are delivered to the process without regard to the instruction that is executing.

The asynchronous signals are SIGHUP, SIGINT, SIGQUIT, SIGALRM, SIGTERM, SIGUSR1, SIGUSR2, SIGHLD, SIGPWR, SIGVTALRM, SIGPROF, SIGIO, SIGWINCH, SIGTSTP, SIGCONT, SIGTTIN, SIGTTOU, and SIGURG.
The Sybase libraries treat the asynchronous signal SIGTRAP as a synchronous signal. See “SIGTRAP Signal” on page 16 for more information.

6.3.4. Signal Handlers

For all UNIX platforms, signal handlers are installed on a per-process basis. In a multithreaded environment, there is only one signal handler for each signal within a process. The last signal handler installed for any thread is valid for all threads in the process. The handler is called when the signal is delivered.

6.3.5. Signal Masking

Signal masking lets you specify that a signal will not be delivered until some condition is met.

Nonthreaded environments have only one thread of control. Each signal is masked or unmasked for the entire process.

In multithreaded environments, signal masking is handled differently on different platforms:

• On platforms that do not support native threads, such as HP 9000/800 HP-UX 10.x, signals are masked on a per-process basis. Changing the signal mask on one thread affects the entire process.

• On platforms that support native threads, such as Sun Solaris 2.x (SPARC), IBM RS/6000 AIX, and others, signals are masked on a per-thread basis. Masking a signal on one thread affects that thread only. To mask a signal for the entire process, you must mask the signal for each of its threads.

Threads spawned by a parent thread inherit the signal mask of the parent thread. You can build applications to take advantage of signal-mask inheritance. If a signal must be masked for an entire process, mask it for the main or initial thread. Any thread created thereafter inherits this thread’s signal mask.

6.3.6. Signal Delivery

A nonthreaded environment has only one thread of control. Synchronous and asynchronous signals are delivered to the process.

In a multithreaded environment, multiple threads represent multiple executing flows of control. A synchronous signal is always delivered
for the thread that caused the exception. An asynchronous signal is delivered for the first executing thread for which delivery of the signal is enabled.

You can specify that an asynchronous signal will be delivered for a thread or set of threads. Unmask the signal for a set of threads to enable signal delivery for these threads. Mask the signal for all other threads to disable signal delivery. The kernel holds a signal until it executes a thread for which delivery of the signal is enabled.

6.3.7. Using the `sigwait()` Call to Handle Asynchronous Signals

In a multithreaded environment, you can use the `sigwait()` UNIX system call to handle asynchronous signals synchronously. This helps your applications avoid interruptions and behave more predictably.

Typically, a dedicated signal-handler thread uses `sigwait()` to wait for any of a set of asynchronous signals. A signal mask provided for `sigwait()` indicates which signals to wait for. When a signal is delivered, `sigwait()` returns the signal number and the signal-handler thread executes the signal handler.

For a process to receive asynchronous signals by means of `sigwait()`, you must:

1. Create a dedicated signal-handler thread in which `sigwait()` is invoked in order to capture the asynchronous signals. Mask these signals for this thread.
2. For all other threads, mask the signals specified in the set argument to `sigwait()`.

   When you mask asynchronous signals in all threads, you make sure that signals are held while a signal-handler thread executes the previous signal’s handler. In subsequent calls to `sigwait()`, the blocked signals are returned.

   If you mask all asynchronous signals at the start of the program, all threads spawned by the main thread inherit this signal mask.

6.3.8. How Version 11.1.x Libraries Handle Signals Internally

Sybase libraries install signal handlers for synchronous signals in the traditional fashion. For asynchronous signal handling, the libraries use a dedicated thread that calls `sigwait()` to wait for signals.
Make sure you use Sybase-provided library calls to install signal handlers. Sybase cannot guarantee results if you install signal handlers using other methods. Also take the following precautions:

- Because Sybase libraries handle asynchronous signals using `sigwait()`, mask asynchronous signals in all of an application’s threads.

  If you mask asynchronous signals or call a Sybase-provided initialization routine such as `ct_init()` or `srv_init()` before spawning any threads, you ensure that asynchronous signals are masked in all threads spawned subsequently.

- Signal handlers for asynchronous signals do not run at an interrupt level. On platforms that do not support native threads, make sure that no thread runs in a tight loop. When a signal occurs, Sybase’s internal signal-handler thread must be able to run and to execute the signal handler.

- Internally, Sybase libraries use the SIGUSR1 signal. Make sure that no other thread in the application or in another library linked to the application calls `sigwait()` for SIGUSR1.

- Some applications may be linked with a Sybase library and with a library that uses `sigwait()` to handle asynchronous signals. In such a case, do not use Sybase-provided calls to install handlers for asynchronous signals handled by the other library.

  In a Sybase library, the signal-handler thread calls `sigwait()` only for SIGUSR1 and for the asynchronous signals it monitors. No conflict occurs if another library monitors asynchronous signals not handled by the Sybase libraries.

### 6.3.9. Special Sybase Signal Handlers

In nonthreaded environments, you can mask or unmask signals using UNIX system calls.

In multithreaded environments, using earlier versions of Open Client/Server, you could not change masking for threads used internally by Sybase libraries. Using version 11.1.1 of the Sybase libraries, however, two special signal handlers are available for masking or unmasking signals:

- `CS_SIGNAL_BLOCK` – To mask a signal, install this signal handler using the Sybase-provided signal installation routine. When the signal occurs, it is held until you unmask it.
• CS_SIGNAL_UNBLOCK – To unmask a signal, install this signal handler using the Sybase-provided signal installation routine. Other special signal handlers for multithreaded environments include:
  • CS_SIGNAL_IGNORE – This signal handler ignores a signal. CS_SIGNAL_IGNORE works the same way as the UNIX special signal handler SIG_IGN.
  • CS_SIGNAL_DEFAULT – This signal handler takes a default action when a signal occurs. CS_SIGNAL_DEFAULT works the same way as the UNIX special signal handler SIG_DFL.

6.3.10. SIGTRAP Signal

Sybase libraries treat the asynchronous signal SIGTRAP as a synchronous signal.

If it were treated as an asynchronous signal, the signal would be masked on the calling thread when an application called srv_init() or ct_init(). That would disable debugging, because many debuggers use SIGTRAP to communicate with the application being debugged. To avoid interfering with debugging, SIGTRAP is treated as a synchronous signal.

6.4. Initialization Error Reporting

Error reporting for initialization routines has changed in Open Client/Server version 11.1.1.

6.4.1. cs_ctx_alloc() and cs_ctx_global()

When an application call to either cs_ctx_alloc() or cs_ctx_global() returns CS_FAIL, extended error information is sent to standard error (STDERR) and to the file sybinit.err. The sybinit.err file is created in the current working directory.

6.4.2. ct_init()

When an application call to ct_init() returns CS_FAIL due to a Net-Library error, extended error information is sent to standard error (STDERR) and to the file sybinit.err. The sybinit.err file is created in the current working directory.
6.5. Context and Connection Properties

New context and connection properties are provided in Open Client/Server version 11.1.1, as shown in Table 7.

Table 7: New context and connection properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Meaning</th>
<th>Possible Values</th>
<th>Applicable Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS_CON_KEEPALIVE</td>
<td>Whether to use the KEEPALIVE option</td>
<td>CS_TRUE or CS_FALSE.</td>
<td>Context or connection</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CS_TRUE is the default.</td>
<td></td>
</tr>
<tr>
<td>CS_CON_TCP_NODELAY</td>
<td>Whether to use the TCP_NODELAY option</td>
<td>CS_TRUE or CS_FALSE.</td>
<td>Context or connection</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CS_TRUE is the default.</td>
<td></td>
</tr>
</tbody>
</table>

Some Net-Library protocol drivers do not support one or both of these properties. After a connection is established on such a protocol driver, calling `ct_con_props()` with CS_GET or CS_SET returns CS_FAIL.

To determine whether a property is supported, an application can call `ct_con_props()` on an established connection. The call must use the CS_SUPPORTED action parameter and must use the buffer parameter as the address of a CS_BOOL variable.

Use an example like this for the CS_CON_KEEPALIVE property:

```c
CS_BOOL boolval;

/*
 ** Is CS_CON_KEEPALIVE supported on this
 ** connection?
 */
ret = ct_con_props(conn, CS_SUPPORTED,
                  CS_CON_KEEPALIVE, &boolval,
                  NULL);
if(ret != CS_SUCCEED)
    ... handle the error ...
    printf("CS_CON_KEEPALIVE %s supported",
           boolval == CS_TRUE ? "is" : "is not");
```
Use an example like this for the CS_CON_TCP_NODELAY property:

```c
CS_BOOL boolval;

/*
** Is CS_CON_TCP_NODELAY supported on this
** connection?
*/
ret = ct_con_props(conn, CS_SUPPORTED,
                   CS_CON_TCP_NODELAY, &boolval,
                   NULL);
if(ret != CS_SUCCEED)
    ... handle the error ...
printf("CS_CON_TCP_NODELAY %s supported",
       boolval == CS_TRUE ? "is" : "is not");
```

### 7. Programming Issues

This section describes programming issues relevant to Open Client/C and Open Server.

#### 7.1. Client-Library

This section describes programming issues specific to Open Client Client-Library version 11.1.1.

##### 7.1.1. `ct_poll`

Do not call `ct_poll` from within any Client-Library callback function or from within any other function that can execute at the system-interrupt level.

Calling `ct_poll` at the system-interrupt level can corrupt Open Client/Server internal resources and cause recursion in the application.

##### 7.1.2. Asynchronous Programming

To properly exit Client-Library, wait until all asynchronous operations are complete, then call `ct_exit`. 
If an asynchronous operation is in progress when `ct_exit` is called, the routine returns `CS_FAIL` and does not exit Client-Library properly, even when `CS_FORCE_EXIT` is used.

### 7.1.3. CS_NOINTERRUPT Connection Property

In the current release of Client-Library, CS_NOINTERRUPT does not prevent notification events from occurring.

### 7.2. DB-Library

This section describes programming issues specific to Open Client DB-Library version 11.1.1.

#### 7.2.1. Recompiling DB-Library 4.x and 10.0.x Applications

Applications written for DB-Library versions 4.x and 10.0.x must be recompiled to use DB-Library version 11.1.1.

#### 7.2.2. Sample Programs and Sample Databases

Examples 3, 9, 10, and 11 require the `pubs2` database to be installed on the server. Example 12 requires the `interpubs` database to be installed on the server. For more details, see the `README` file in the `$SYBASE/sample/dblibrary` directory.

### 8. Utility Issues

This section describes issues that concern the standalone utilities.

#### 8.1. Utility Messages

With this release, messages generated by the `bcp`, `defncopy`, and `isql` utilities have changed. If you process these messages with scripts that parse specific strings (such as with `awk` or `grep`), you may need to change the search patterns of the scripts to accommodate the new messages.
9. Technical Support

Each Sybase installation that has purchased a support contract has one or more designated people who are authorized to contact Sybase Technical Support. If you have any questions about this installation or if you need assistance during the installation process, ask the designated person to contact Sybase Technical Support or the Sybase subsidiary in your area.

10. Other Sources of Information

Use the Sybase Technical Library CD and the Technical Library Web site to learn more about your product:

- Technical Library CD contains product manuals and technical documents and is included with your software.
- Technical Library Web site includes the Product Manuals site, which is an HTML version of the Technical Library CD that you can access using a standard Web browser. In addition, you’ll find links to the Technical Documents Web site (formerly known as Tech Info Library), the Solved Cases page, and Sybase/Powersoft newsgroups.

To access the Technical Library Web site, go to support.sybase.com, click the Electronic Support Services tab, and select a link under the Technical Library heading.

10.1. Sybase Certifications on the Web

Technical documentation at the Sybase Web site is updated frequently.

For the latest information on product certifications and/or the EBF Rollups:

1. Point your Web browser to Technical Documents at the following Web site:
   techinfo.sybase.com
2. In the Browse section, click on the What’s Hot entry.
3. Explore your area of interest: Hot Docs covering various topics, or Hot Links to Technical News, Certification Reports, Partner Certifications, and so on.
If you are a registered SupportPlus user:

1. Point your Web browser to Technical Documents at the following Web site:
   techinfo.sybase.com
2. In the Browse section, click on the What’s Hot entry.
3. Click on the EBF Rollups entry.
   You can research EBFs using Technical Documents, and you can download EBFs using Electronic Software Distribution (ESD).
4. Follow the instructions associated with the SupportPlus Online Services entries.

If you are not a registered SupportPlus user and you want to become one:

You can register by following the instructions on the Web.

To use SupportPlus, you need:

- A Web browser that supports the Secure Sockets Layer (SSL), such as Netscape Navigator 1.2 or later
- An active support license
- A named technical support contact
- Your user ID and password

Whether or not you are a registered SupportPlus user:

You can use Sybase’s Technical Documents. Certification Reports are among the features documented at this site.

1. Point your Web browser to Technical Documents at the following Web site:
   techinfo.sybase.com
2. In the Browse section, click on the What’s Hot entry.
3. Click on the topic that interests you.