Release Bulletin
Adaptive Server Enterprise Version 12.0 for Sun Solaris

Document ID: 73430-01-1200-08

Last revised: August 2001

Topic	Page
1. Accessing current release bulletin information	1
2. Component summary	2
3. Additional requirements	5
4. Known installation and upgrade issues	7
5. Special installation instructions	8
6. Special upgrade instructions	9
7. New functionality in this version	17
8. Changed functionality in this version	17
9. Changes affecting applications after upgrade	23
10. Known problems	26
11. Documentation updates and clarifications	35
12. Technical support	65
13. Other sources of information	65

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 Manuals Web site.

❖ **To access release bulletins at the Technical Library Product Manuals Web site:**

1 Go to Product Manuals at http://www.sybase.com/support/manuals/.

2 Follow the links to the appropriate Sybase product.

3 Select the "Platform-Specific" collection for the product version you are interested in.

4 From the list of individual documents, select the link to the release bulletin for your platform. You can either download the PDF version or browse the document online.

2. Component summary

Enclosed is Sybase® Adaptive Server® Enterprise version 12.0. Server and client components are distributed on separate CDs.

For details on system requirements, including disk space and RAM, see the *Installation Guide* for your platform.

For more information specific to jConnect™, jisql, and Ribo, see the jConnect documentation.

This release bulletin provides the latest information about and known problems with:

- Studio Installer
- Upgrade on Windows NT
- sqlupgrade and sqlupgraderes
- svrbuild and srvbuildres

2.1 Installation kit

The installation kit includes:

- The server CD – for contents list, see "Server components" on page 3.
- The PC-client CD – contains software client components to be installed on Windows 95, Windows 98, and Windows NT computers.
- *Release Bulletin Adaptive Server Enterprise Version 12.0 for Sun Solaris*
- *Installation Guide Sybase Adaptive Server Enterprise Version 12.0 Sun Solaris*

2.2 Server components

The Server CD includes the following components:

- Adaptive Server 12.0 (includes Backup Server)
- Adaptive Server Enterprise Monitor™ Server 12.0
- Monitor Client Library 12.0
- Historical Server 12.0
- 11.5.1 Monitor Server for 11.0.x
- Language Modules 12.0
- jConnect for JDBC 4.2
- jConnect for JDBC 5.2
- Java utilities:
  - jisql 2.0
  - Ribo 2.0
  - Cascade 1.0
- SQL Remote™ 6.0.2
- Open Client™ 12.0 SDK
- Embedded SQL™/C 12.0 SDK
- Embedded SQL/Cobol 12.0 SDK
- Standard Full Text Search Specialty Data Store 12.0

2.3 PC-client components

The PC-client CD includes the following components, which can be installed on Windows 95, Windows 98, and Windows NT client computers:

- Sybase Central™ 3.2
- Adaptive Server Enterprise plug-in for Sybase Central 12.0
- InfoMaker® 7.0.1
- jConnect 4.2 and related documentation
- jConnect 5.2 and related documentation
- Java utilities:

  - jisql 2.0
  - Ribo 2.0
  - Cascade 1.0
- Language Modules 12.0
- Open Client 12.0 SDK
- Open Client Runtime 12.0
- XA Interface Library for ASE Distributed Data Manager 12.0
- ODBC 3.5.1
- OLEDB 2.1
- PowerDynamo™ 3.0
- SQL Modeler™ 6.1.4
- SQL Remote 6.0.2
- Monitor Client Library 12.0
- Embedded SQL/C 12.0 SDK
- Embedded SQL/Cobol 12.0 SDK

Note Java Central is installed with PowerDynamo, not Adaptive Server. The only plug-in that comes with Adaptive Server is the Adaptive Server Enterprise plug-in for Sybase Central 12.0.

2.4 Component compatibility

This section lists the Sybase components that are compatible with Adaptive Server Enterprise version 12.0. For information about operating system requirements, see individual component documentation.

The following components have been tested for compatibility with Adaptive Server 12.0:

- Sybase Character Sets 3.0
- Sybase Central viewer 3.2
- DirectConnect™ Anywhere 11.7
- DirectConnect for Informix 11.7

- DirectConnect for MVS 11.7
- DirectConnect for Oracle 12.0
- jConnect for JDBC (all versions)
- Open Client/Open Server™ 12.0

  OpenClient 11.1.1 is compatible with Adaptive Server 12.0 except it does not support transparent client reconnection in failover in an HA environment
- ODBC Driver for Adaptive Server 12.0
- InfoMaker 7.0.1
- Replication Server® 11.5.1, 12.0
- Replication Agent™ for Adaptive Server Anywhere 6.0.0
- Replication Agent for DB2 12.0
- Replication Agents for Informix, MS SQL Server, and Oracle 12.0
- Security Guardian™ 11.1.1
- SQL Remote 6.0.2
- SQL Modeler 6.1.4
- XA-Library™ for CICS/ENCINA 11.1.1
- XA-Server™ for Tuxedo 11.1.1

Note Backup Server 12.0 has been tested compatible with Backup Server 11.9.2.1. Either server can be local or remote. Backup Server 12.0 is compatible with versions of Backup Server prior to 11.9.2.1 *only* if the Backup Server 12.0 is the remote server.

3. Additional requirements

This section describes requirements for using Adaptive Server in addition to any requirements specified in the published documentation.

3.1 Operating system patches

Solaris 2.6 (32-bit) requires the following operating system patches to run Adaptive Server 12.0 components:

- 105284-16
- 105490-06 (memory leak patch)
- 105181-12, 105568-13 (JRE patches)
- 105181-06
- 105210-09
- 105529-03
- 105786-05
- 104668-09 (c 4.2)
- 104631-07 (c++ 4.2)

Solaris 2.7 (64-bit) requires the following operating system patches to run Adaptive Server 12.0 components:

- 106541-03
- 106327-05
- 106300-06

If your operating system requires patches, install them before you install Adaptive Server components.

To determine which patches have been installed on your system, see "Viewing installed patches" on page 6.

Contact your operating system provider for any patches recommended for your installation. Do not use a patch that is earlier than the version suggested for your operating system. Use the patch recommended by the operating system vendor, even if it supersedes the patch listed.

If you plan to use Sybase Failover in a high availability system, install all vendor-recommended operating system and high availability patches. For more information, contact your operating system and high availability vendors.

3.1.1 Viewing installed patches

To list all currently installed patches and display the operating system version level, enter:

```
showrev -p
```

4. Known installation and upgrade issues

This section describes known problems or issues that you may encounter during the installation or upgrade process.

4.1 Studio Installer fails to load on Pentium 4 machines

Studio Installer does not work on Pentium 4 machines running any Windows operating systems. An error occurs in *java.exe* when the Studio Installer attempts to install Sybase products on a Pentium 4 machine. The error message may be similar to one of the following:

On Dr. Watson for Windows NT:

```
An application error has occurred and an application
error log is being generated. java.exe Exception: access
violation90xc0000005), Address: 0x500bf974
```

On MS-DOS:

```
Error: java.lang.ExceptionInInitializerError

java.lang.ExceptionInInitializerError
at java.util.TimeZone.getDefault<TimeZone.java:192>
at
at Logfile.printIn<Logfile.java:148>
at Installer.main<Compiled Code>
```

For a list of the products affected, a full description of the problem, and a workaround, see the Sybase Web site at http://www.sybase.com/detail/1,3693,1013241,00.html.

4.2 Input cursor not always visible in Studio Installer

Under some circumstances no input cursor is visible in the text field of the dialog box. This error is caused by interactions between the Java libraries and the platform-specific Windows libraries. Usually, it is not a problem if you are directly connected to the system on which you are running the installer. The cursor is set to the first text field, so when you start typing, the characters are entered into that field.

Workaround: If no cursor is visible, click in the field in the dialog box before entering characters. Alternatively, you can minimize the Studio Installer window, then maximize it.

See Bug #204713.

4.3 Erroneous message when upgrading from 11.0.x servers

When upgrading from an 11.0.x SQL Server,® you may see a message similar to the following:

```
Wed Sep 29 11:47:48 1999: Sybsystemprocs has size of 21
MB. It requires a minimum of 80 MB for release 12.0.

Database sybsystemprocs requires approximately 3 MB
more free space in order to be upgraded.
```

Workaround: Ignore the second paragraph of the message that indicates the amount of free space required by sybsystemprocs. The suggested number is incorrect.

See Bug #207073.

4.4 Using Adaptive Server with transaction services

To use Adaptive Server with a transaction monitor such as Tuxedo, Encina, or Microsoft MTS/DTC, you must:

- Obtain a license for DTM use.
- Configure Adaptive Server for DTM.
- Give the user dtm_tm_role.

See *Using Adaptive Server Distributed Transaction Management Features* for more information.

5. Special installation instructions

This section provides information for installing Adaptive Server 12.0 that was not included or should be corrected in the *Installation Guide*.

5.1 Correction to unloading procedure

In the Adaptive Server 12.0 *Installation Guide*, Chapter 3, “Unloading Server Products from Distribution Media,” in the section “Unloading Components with Studio Installer: GUI Mode,” ignore the following paragraph (sixth down from the top):

"During the installation process, the computer must be restarted. After restarting, the computer accesses the CD for additional data. For this reason, the CD must remain in the CD-ROM drive during restart, and the CD-ROM drive should be on the installation machine."

This information does not apply to UNIX platforms.

5.2 Install *hasybase* agent

If you are configuring Adaptive Server as a companion server in a high availability system, you must install Sun's hasybase agent before you install Adaptive Server.

6. Special upgrade instructions

This section provides information for upgrading to Adaptive Server 12.0 that was not included in the *Installation Guide*.

6.1 *dbcc upgrade_object* issues

Adaptive Server version 11.9.3 introduced the process of upgrading compiled objects based on their source text. Compiled objects are:

- Check constraints
- Defaults
- Rules
- Stored procedures (including extended stored procedures)
- Triggers
- Views

The source text of each compiled object is stored in the syscomments table, unless it has been manually deleted. When you upgrade the server, the existence of the source text in syscomments is verified during that process. However, the compiled objects are not actually upgraded until they are invoked.

For example, say you have a user-defined stored procedure named list_proc. The presence of source text for list_proc is verified when you upgrade to Adaptive Server 11.9.3. Then, the first time list_proc is invoked after the upgrade, Adaptive Server detects that the list_proc compiled object has not been upgraded. Adaptive Server recompiles list_proc, based on the source text in syscomments. The newly compiled object is then executed.

Upgraded objects retain the same object ID and permissions that they used prior to being upgraded.

Compiled objects for which the source text was hidden using sp_hidetext are upgraded like objects for which the source text is not hidden. For information on sp_hidetext, see the *Adaptive Server Reference Manual*.

To ensure that compiled objects have been upgraded successfully *before* they are invoked, you can upgrade them manually using the dbcc upgrade_object command. For details, see "Finding compiled object errors before production."

6.1.1 Finding compiled object errors before production

Changes made in previous versions of Adaptive Server may cause compiled objects to work differently in version 11.9.3 and later. You can use dbcc upgrade_object to find the following errors and potential problem areas that may require manual changes to achieve the correct behavior:

- Reserved word errors
- Missing, truncated, or corrupted source text
- Quoted identifier errors
- Temporary table references
- select * potential problem areas

After reviewing the errors and potential problem areas, and fixing those that need to be changed, you can use dbcc upgrade_object to upgrade compiled objects manually instead of waiting for the server to upgrade the objects automatically. For details, see "Using dbcc upgrade_object" on page 14.

Reserved word errors

If dbcc upgrade_object finds a reserved word used as an object name in a compiled object, it returns an error, and the upgrade of that object fails. To fix the error, either manually change the object name or use quotes around the object name and issue the command set quoted identifiers on. Then, re-create the compiled object.

For example, suppose you load a database dump from Adaptive Server 11.5 into Adaptive Server 11.9.3 and the dump contains a stored procedure that uses the word "lock." When you run dbcc upgrade_object on that stored procedure, the command returns an error because, although "lock" was not reserved in version 11.5, it became a reserved word in version 11.9.2. With this advance notice, you can change the stored procedure and any related tables before they are used in a production environment.

Missing, truncated, or corrupted source text

If the source text in syscomments was deleted, truncated, or otherwise corrupted, dbcc upgrade_object may report syntax errors. If the source text was not hidden, you can use sp_helptext to verify the completeness of the source text. If truncation or other corruption has occurred, drop and re-create the compiled object.

Quoted identifier errors

dbcc upgrade_object returns a quoted identifier error if:

- The compiled object was created in a pre-11.9.2 version with quoted identifiers active (set quoted identifiers on).
- Quoted identifiers are not active (set quoted identifiers off) in the current database.

To avoid this error, activate quoted identifiers before running dbcc upgrade_object. When quoted identifiers are active, you must use single quotes instead of double quotes around quoted dbcc upgrade_object keywords.

If quoted identifier errors occur, use the set command to activate quoted identifiers, and then run dbcc upgrade_object to upgrade the object.

For compiled objects created in 11.9.2, the upgrade process automatically activates or deactivates quoted identifiers as appropriate.

Note Quoted identifiers are not the same as literals enclosed in double quotes. The latter do not require you to perform any special action before the upgrade.

Temporary table references

If a compiled object such as a stored procedure or trigger refers to a temporary table (#temp *table_name*), which was created outside the body of the object, the upgrade fails, and dbcc upgrade_object returns an error. To correct this error, create the temporary table exactly as expected by the compiled object, and then execute dbcc upgrade_object again. This is not necessary if the compiled object is upgraded automatically when it is invoked.

select * potential problem areas

In Adaptive Server version 11.9.3 and later , the results of a select * clause in a stored procedure (created before version 12.0), trigger, or view that was created in an earlier version of Adaptive Server may be different from what you expect. For more information about the changes, see "Changes to results returned by select *" on page 12.

If dbcc upgrade_object finds a select * clause in the outermost query block of a stored procedure, it returns an error, and does not upgrade the object.

For example, if a stored procedure has the following statements:

```
create procedure myproc as
  select * from employees
create procedure yourproc as
  if exists (select * from employees)
    print "Found one!"
```

dbcc upgrade_object returns an error on myproc because myproc includes a statement with a select * clause in the outermost query block. This procedure is not upgraded.

dbcc upgrade_object does not return an error on yourproc because the select * clause occurs in a subquery. This procedure is upgraded.

Changes to results returned by *select* *

In Adaptive Server 11.9.3, select * in a stored procedure, trigger, or view that was created in a pre-11.9.3 version of Adaptive Server may return different results than expected. The following new behaviors may occur:

- If users do not have correct permissions on all columns of the table, execution of the stored procedure, trigger, or view could fail.
- If users have correct permissions, they see all columns in the table, including those added with alter table after the stored procedure, trigger, or view was created.
- Cursors containing a select * statement may return different results than expected.

To avoid these problems, run the dbcc upgrade_object command to find occurrences of select *. Then, confirm for each occurrence, that the retrieval of all columns in the specified table provides the correct results. You may need to change the select * statement to a select statement that specifies only the column names you want to retrieve. For details on using dbcc upgrade_object, see "Finding compiled object errors before production" on page 10.

Determining whether *select* * should be changed in views

If dbcc upgrade_object reports the existence of select * in a view, compare the output of syscolumns for the original view to the output of the table, to determine whether columns have been added to or deleted from the table since the view was created.

For example, suppose you have the following statement:

```
create view all_emps as select * from employees
```

Before upgrading the all_emps view, use the following queries to determine the number of columns in the original view and the number of columns in the updated table:

```
select name from syscolumns
  where id = object_id("all_emps")
select name from syscolumns
  where id = object_id("employees")
```

Compare the output of the two queries. If the table contains more columns than the view, and retaining the pre-upgrade results of the select * statement is important, change the select * statement to a select statement with specific column names. If the view was created from multiple tables, check the columns in all tables that comprise the view and rewrite the select statement if necessary.

Warning! Do not execute a select * statement from the view. Doing so will upgrade the view and the information about the original column information in syscolumns will be overwritten.

Another way to determine the difference between the columns in the view and in the new tables is to run sp_help on both the view and the tables that comprise the view.

This comparison works only for views, not for other compiled objects. To determine whether select * statements in other compiled objects need to be revised, review the source text of each compiled object.

6.1.2 Using dbcc upgrade_object

Syntax

```
dbcc upgrade_object [ ( dbid | dbname
[, ['database.[owner].]compiled_object_name' |
 'check' | 'default' | 'procedure' | 'rule' |
 'trigger' | 'view'
 [, 'force' ] ] ) ]
```

where:

- *dbid* specifies the database ID. If you do not specify *dbid*, all compiled objects in the current database are upgraded.
- *dbname* specifies the database name. If you do not specify *dbname*, all compiled objects in the current database are upgraded.
- *compiled_object_name* is the name of a specific compiled object you want to upgrade. If you use the fully qualified name, *dbname* and *database* must match, and you must enclose the fully qualified name in quotes. If the database contains more than one compiled object of the same name, use the fully qualified name. Otherwise, all objects with the same name are parsed, and if no errors are found, upgraded.
- check upgrades all check constraints and rules. Referential constraints are not compiled objects and do not require upgrading.
- default upgrades all declarative defaults and the defaults created with the create default command.
- procedure upgrades all stored procedures.
- rule upgrades all rules and check constraints.
- trigger upgrades all triggers.
- view upgrades all views.

The keywords check, default, procedure, rule, trigger, and view specify the classes of compiled objects to be upgraded. When you specify a class, all objects in that class, in the specified database, are upgraded, provided that dbcc upgrade_object finds no errors or potential problem areas.

- force specifies that you want to upgrade the specified object even if it contains a select * clause. Do not use force unless you have confirmed that the select * statement will not return unexpected results. The force option does not upgrade objects that contain reserved words, contain truncated or missing source text, refer to nonexistent temporary tables, or do not match the quoted identifier setting. These objects must be fixed before they can be upgraded.

Note If set quoted identifiers is on, use single quotes around the keywords. If set quoted identifiers is off, you can use either double quotes or single quotes.

Examples

```
dbcc upgrade_object
```

Upgrades all compiled objects in the active database.

```
dbcc upgrade_object(listdb, 'procedure')
```

Upgrades all stored procedures in the listdb database. Single quotes are used around procedure because set quoted identifiers is on.

```
dbcc upgrade_object(listdb, "rule")
```

Upgrades all rules and check constraints in the listdb database. Double quotes are used around rule because set quoted identifiers is off.

```
dbcc upgrade_object(listdb, list_proc)
```

Upgrades all stored procedures named list_proc in the listdb database.

```
dbcc upgrade_object(listdb,
  "listdb.jkarrik.list_proc")
```

Upgrades the stored procedure list_proc, which is owned by the login "jkarrik".

```
dbcc upgrade_object(master,
  "listdb.jkarrik.list_proc")
```

Returns an error because the value of *dbname* is master and the value of *database* is listdb. These values must match.

Permissions

Only the Database Owner or a System Administrator can execute dbcc upgrade_object. The Database Owner can upgrade his or her own objects in the database.

Increasing the log segment size

You can specify that all compiled objects of a particular class should be upgraded in one execution of dbcc upgrade_object; for example, you can upgrade all triggers by using the trigger keyword. However, even though you use only one dbcc command, the upgrade of each object is recorded in a separate transaction; the old row is deleted from sysprocedures and a new row is written. Therefore, if you run dbcc upgrade_object on a large number of compiled objects, your system may run out of log space. Be sure to increase the size of the log segment in the databases in which you plan to run this command, to allow sufficient room to log all the upgrades.

Error reporting

To send all the output from dbcc upgrade_object to the screen, a System Administrator can start the server with dbcc traceon(3604). Using this command is recommended if you think the output of error messages might overflow the error log.

6.1.3 Upgrading compiled objects in database dumps

When you load a database dump that was created in an earlier version than the current Adaptive Server, you are not required to perform the pre-upgrade tasks before loading the dump. Therefore, you will not receive any notification if the compiled objects in your database dump are missing their source text. After loading a database dump, run sp_checksource to verify the existence of the source text for all compiled objects in the database. Then, you can allow the compiled objects to be upgraded as they are executed, or you can run dbcc upgrade_object to find potential problems and upgrade objects manually.

For information on using sp_checksource, see the *Adaptive Server Reference Manual*.

6.1.4 Determining whether a compiled object has been upgraded

To determine whether a compiled object has been upgraded, do one of the following:

- Look at the sysprocedures.version column. If the object was upgraded, this column will contain the number 12000.
- If you are upgrading to a 64-bit pointer size in the same version, look at the sysprocedures.status column. It will contain a hexadecimal bit setting of 0x2 to indicate that the object uses 64-bit pointers. If the bit is not set, the object is a 32-bit object, which means it was not upgraded.

7. New functionality in this version

This section describes new functionality in the Studio Installer and Adaptive Server 12.0.

7.1 Studio Installer

In this version, the Studio Installer replaces sybsetup. Use the Studio Installer to:

- Unload and install all Sybase components using a consistent installation interface.
- Install only commonly installed components from the distribution media (Standard installation type).
- Install specific components (Custom installation type).
- Install all components (Full installation type). During installation, you can choose to overwrite existing components.
- Configure components either at the time of installation or later using Sybase installation and configuration utilities.
- Install multiple versions of components, such as Open Client and jConnect.

7.2 Adaptive Server 12.0

For information about new Adaptive Server 12.0 features, see *What's New in Adaptive Server 12.0?*

8. Changed functionality in this version

This section describes changes related to installation.

8.1 Installation tool requirements

The Studio Installer is Java-based and uses XML input, which makes the installation process similar on both UNIX-based and Windows-based computers. The Studio Installer requires a window package to be installed to use the graphical user interface (GUI).

The work of the installer consists of several steps. After you have completed the interactive dialogue, the installer creates the target directory, if necessary, and unloads all of the selected components into the target directory. When this is complete, you can configure the components. Not every component has a configuration utility. Configuration of components is optional; you can skip this activity during the installation and perform them at a later time.

Licensing of components is done through Sybase Software Asset Management (SySAM), which is a licensing mechanism that provides System Administrators with a means to monitor their site's use of registered Sybase products and optional features. SySAM handles check-in and check-out requests from all other Adaptive Server instances in the system. For more information about using SySAM, see Chapter 4, "Sybase Software Asset Management" in the *Installation Guide* for your platform.

8.2 Directory structure

Most components, including the executable program, installation and configuration tools, and display-related files needed by the component, are installed in their own subdirectories. The naming convention for subdirectories includes a component identifier, such as ASE or jConnect, and the software release version, such as 12_0 or 4_2.

For example, Adaptive Server is installed in *$SYBASE/ASE-12_0*. Shared components are installed in subdirectories that are separate from component subdirectories. For example, Open Client is installed in *$SYBASE/OCS-12_0*.

This directory structure enables you to install into an existing *$SYBASE* directory structure, and to install and use multiple versions of some components.

For more information about the new directory structure, see the *Installation Guide* for your platform.

8.3 DECNet not supported

The DECNet network protocol is not supported in Adaptive Server 12.0.

8.4 *dbid*s for some system databases start at number 31513

In Adaptive Server 12.0, dbids for the following databases start at 31513:

- dbccalt

- dbccdb
- sybsecurity
- sybsystemdb
- sybsystemprocs

If you drop and re-create these databases after an upgrade, Adaptive Server applies these new dbids. The dbids for all other databases are determined in the same way as in earlier versions.

8.5 More stripes available for Backup Server

In earlier versions of Backup Server, the maximum number of stripes you could use to dump a database to disk or tape was 32. In Backup Server 12.0, more stripes are available.

8.5.1 Maximum stripes, network connections, and file descriptors

The maximum number of stripes that Backup Server can use is limited by the maximum number of Open Server threads it can create. Open Server imposes a maximum limit of SRV_MAX_SRVPROCS (12k) on the number of threads an application can create.

Backup Server creates one service thread for each stripe. Therefore, the maximum number of local stripes Backup Server can dump to or load from is 12,288 (`SRV_MAX_SRVPROCS = 12288`).

The maximum number of network connections a local Backup Server can originate is limited by Open Server to 9118. Therefore, the maximum number of remote stripes that Backup Server can use in a single dump or load operation is 9118.

A remote Backup Server can accept a maximum of 4096 server connections into it at any given time. Therefore, the maximum number of remote stripes to a single remote Backup Server is 4096. Open Server and the operating system also impose limits on the number of file descriptors.

Backup Server uses two file descriptors for each stripe apart from the file descriptors associated with the error log file, *interfaces* file and other system files. However, there is a per-thread limitation imposed by the operating system on the number of file descriptors. Open Server has a limitation of 1280 on the number of file descriptors that an application can keep track of.

The formula for determining the approximate maximum number of local stripes to which Backup Server can dump is:

```
[ min(OS limitation, OpenServer limitation) - 2 ] / 2
```

The formula for determining the approximate maximum number of remote stripes to which Backup Server can dump is:

```
[ min(OS limitation, OpenServer limitation) - 2 ] / 3
```

For details about the default and maximum file descriptor limits, see your operating system documentation.

8.5.2 Configuration issues

When you start Backup Server, use the -P flag to configure the Open Server for the maximum number of threads it will create. The maximum number of threads equals the maximum number of stripes available. If you have started Backup Server without setting a high enough -P value and you attempt to dump or load a database to a number of stripes that exceeds the number of threads, the dump or load operation will fail.

You must configure the local and remote Backup Servers at start-up by providing the appropriate values to the command-line options. A remote dump to greater than 25 stripes with the local and remote Backup Serverss started with default configuration will fail because the maximum number of network connections that Backup Server can originate (specified by the -N option) is by default 25. The maximum number of server connections into the remote Backup Server (specified by the -C option) is 30.

To configure the system to use the higher stripe limitations, set the following operating system parameters:

- Number of shared memory segments
- Number of shared memory identifiers
- Swap space

If these parameters are not configured properly, when a dump is started to (or a load is started from) a large number of stripes, the operation may abort because of lack of system resources. In this case, you receive a message that Backup Server could not create or attach to a shared memory segment and therefore the SYBMULTBUF processes are terminated.

For more information, see your operating system documentation and "Improving Dump or Load Performance" in Chapter 27, "Backing Up and Restoring User Databases," in the *System Administration Guide*.

8.5.3 Label changes

In earlier versions of Backup Server, the stripe number was stored in the HDR1 label in ASCII format in 4 bytes. In Backup Server 12.0 the stripe number must be stored in integer format. Earlier versions of Backup Server will not be able to load a dump file that uses the version 12.0 dump format. However, Backup Server 12.0 can read and write earlier versions of the dump format.

When performing a dump or load operation involving one or more remote servers, if one or more of the remote servers use the label format of an earlier version, and the number of stripes to which the database is dumped (or from which it is loaded) is greater than 32, then the operation aborts with an error message.

8.6 Sybase Failover does not support XA

Adaptive Servers configured with Sybase Failover in a high availability system do not support pre-12.0 XA or XA-Server.

8.7 Sybase Failover supports only one server per node

Sybase Failover supports only one Adaptive Server per node.

8.8 Dropping proxy databases in a high availability system

You cannot drop proxy databases that have names similar to __pxy from Adaptive Servers configured with Sybase Failover in a high availability system. If you attempt to drop these proxy databases, Adaptive Server issues an error message similar to the following:

```
database 'database_name' has not been recovered yet -
please wait and try again
```

This is expected behavior.

8.9 libhas1.so no longer needed for Sybase Failover

The Beta release of Sybase Failover for Adaptive Server 12.0 required the *libhas1.so* library. This library is no longer needed.

8.10 Concrete identification

Adaptive Server version 12.0 makes changes in the way that ownership chains are resolved for procedures, views and triggers. This may require changes in applications where user-name aliasing is used for creators of database procedures, views, and triggers that reference other objects, particularly when there are cross-database references. Adaptive Server identifies users during a session by login name. This identification applies to all databases in the server. When the user creates an object, the server associates both the owner's database user ID (*uid*) and the creator's login name with the object in the sysobjects table. This information concretely identifies the object as belonging to that user, which allows the server to recognize when permissions on the object can be granted implicitly.

8.10.1 Changes to *sysobjects*

The new loginame column in the sysobjects table contains the server login name of the user who created the object. If the database owner created the object, the name is null.

Identification of pre-12.0 objects

Objects created in versions of Adaptive Server earlier than 12.0 have a null entry in sysobjects.loginame. For these objects, Adaptive Server uses the pre-12.0 behavior for managing object permissions.

8.10.2 How implicit permissions work

If an Adaptive Server user creates a table and then creates a procedure that accesses the table, any user who is granted permission to execute the procedure does not need permission to access the object directly. For example, by giving user "mary" permission on proc1, she can see the id and descr columns from table1, though she does not have explicit select permission on the table:

```
create table table1 (id     int,
                     amount money,
                     descr    varchar(100))

create procedure proc1 as select id, descr from table1

grant execute on proc1 to mary
```

There are, however, some cases where implicit permissions are only useful if the objects can be concretely identified. One case is where aliases and cross-database object access are both involved.

8.10.3 Impact on existing applications

When you upgrade to Adaptive Server 12.0, there is *no effect* on the permissions for any existing (created in versions prior to 12.0) procedures, views, or triggers that reference existing tables. However, if these objects are dropped and re-created, Adaptive Server 12.0 concrete identification is used. Any new procedures, views and triggers use concrete identification. In general, you may want to avoid having objects owned by aliased users. If user-name aliases create problems, having the affected objects owned by the DBO solves permission and name resolutions problems.

8.10.4 Dropping an alias

You cannot drop an alias if the aliased login created any objects or thresholds. Before using sp_dropalias to remove an alias that has performed these actions, remove the objects or procedures. If you still need them after dropping the alias, recreate them with a different owner.

9. Changes affecting applications after upgrade

This section describes system changes introduced by Adaptive Server that may affect your applications if you are upgrading from a previous version.

9.1 New directory structure and environment variables

In this version, Adaptive Server introduces a new directory structure and new environment variables to reference Adaptive Server and Open Client/Server installation directories. Update all scripts that reference the old *$SYBASE* directory structure to use the new paths and environment variables. For more information, see the *Installation Guide* for your platform.

9.2 Procedure text required for upgrade

To upgrade system procedures, the system procedure text must be available in syscomments. For more information, see “Pre-Upgrade Tasks” in the *Installation Guide* for your platform.

9.3 dsync option *on* by default for database device files

The dsync option is used with the disk init and disk reinit commands. By default, Adaptive Server enables dsync for database device files. This option ensures that Adaptive Server can recover data from devices on file systems. However, dsync can cause a degradation in performance for device files that experience high write activity.

When you install a new 12.0 Adaptive Server, by default dsync is set on for all devices.

When you upgrade a UNIX server that stores databases on UNIX file system devices, by default dsync is set:

- on for the master device
- off for all other devices.

For more information about dsync, see "Updates to disk init and disk reinit" on page 46. See also, sp_deviceattr, and sp_helpdevice in the *Adaptive Server Reference Manual*.

9.4 Database to support DTM features

The sybsystemdb database is required to support new distributed transaction management features. Before installation, make sure you have enough space available on the default segment to support sybsystemdb.

For more information, see *Using Adaptive Server Distributed Transaction Management Features*.

9.5 Increased optimization time for queries with many join keys

Queries having long chains of join keys may require additional time to optimize with Adaptive Server. If the time required to optimize such a query is unacceptable, consider using an abstract query plan for the query. For more information, see the *Performance and Tuning Guide.*

9.6 Adaptive Server plug-in to Sybase Central lists fewer servers

The Adaptive Server plug-in to Sybase Central no longer displays all servers listed in the *sql.ini* file. Instead, Sybase Central lists only those servers that you connected to earlier, or those servers that are started as Windows NT services.

To access a new server for the first time, use the Connect menu option to select a server listed in the *sql.ini* file.

9.7 ANSI outer join syntax recommended

Adaptive Server provides support for ANSI outer joins. Sybase recommends that applications make use of the ANSI outer join syntax. For more information, see the *Transact-SQL User's Guide*.

9.8 *lock spinlock ratio* parameter replaced

Adaptive Server 12.0 includes the following new configuration parameters to provide more control over spinlock ratio settings:

- lock address spinlock ratio controls the spinlock ratio for address locks.
- lock table spinlock ratio controls the spinlock ratio for table locks.

9.9 *data_pgs* command includes new *dbid* parameter

The data_pgs command includes a new optional parameter, *dbid*. Any stored procedures that use data_pgs must be re-created.

The new syntax is:

```
data_pgs([dbid], object_id, {data_oam_pg_id | index_oam_pg_id}
```

Where:

- dbid – is the dbid of the database that contains the data pages.
- The definitions for *object_id*, *data_oam_pg_id*, and *index_oam_pg_id* are in the *Adaptive Server Reference Manual*.

For example:

```
select o.name,
Pages = data_pgs(4,i.id, i.doampg)
from pubs2..sysindexes i, pubs2.sysobjects o
where i.id = o.id
and i.id > 100
and (indid = 1 or indid = 0)
```

For more information about the data_pgs command, see the *Adaptive Server Reference Manual*.

10. Known problems

The following sections describe known problems and workarounds for Adaptive Server.

10.1 Asynchronous I/O in UFS, large pools, and named caches

When using asynchronous I/O in UFS on Solaris 2.6 and using Adaptive Server named caches with buffer pools greater than 2K, if Adaptive Server crashes or is shut down using the with no wait option under heavy I/O conditions, the pages on disk may be corrupted.

Note This error is caused by a Sun operating system problem. The Sun service order number is 4226801. When Sun corrects the problem, the workarounds will not be necessary.

Workarounds:

- Do not use asynchronous I/O with UFS on Solaris. Instead, use asynchronous I/O with raw partitions, or
- Use Adaptive Server named caches only with 2K buffer pools.

See Bug # 207482.

10.2 Rebuilding system databases on upgraded servers

Warning! Before you use this procedure, download ESD#1 either from the Sybase Web site at www.sybase.com or from the ESD#1 CD-ROM.

The procedure described in this section is needed only for a server that has been upgraded to 12.0 from an earlier server version. This procedure replaces the procedure in the *Adaptive Server Troubleshooting and Error Messages Guide* for situations when the master database is corrupt, the master device is not affected, and one or both of the following conditions are true:

- There is no current backup of the master database.
- One or more user databases use space on the master device, and there are no current backups of those databases.

If you have current dumps of the master database and all databases that occupy space on the master device, you can use 12.0 buildmaster to build a new master device. You can then load your dump of the master database and your database dumps.

buildmaster creates the master, model, and tempdb databases at particular locations on the master device. The 12.0 buildmaster program changes these locations from previous versions, so you cannot use 12.0 buildmaster to rebuild the system databases on a server that has been upgraded from a previous version. This affects only upgraded installations, not new servers created using version 12.0.

If you are uncertain of the buildmaster version used for your server, check the server's error log, which contains one of these two messages:

```
This installation was created using a pre-12.0 version
of buildmaster.
```

or

```
This installation was created using a 12.0 or later
version of buildmaster.
```

Note Due to the complexity and risk of the following workaround, Sybase strongly recommends that you dump the master database and all databases that use space on the master device after you complete an upgrade to 12.0. You should dump these databases regularly.

Workaround: After upgrading to Adaptive Server 12.0, if you need to rebuild the master or model databases because of corruption, use the 11x_bmaster from the ESD#1 distribution, *not* 12.0 buildmaster. To rebuild master, see "Rebuilding the master database" on page 27. To rebuild model see, "Rebuilding the model database" on page 29.

10.2.1 Rebuilding the master database

During this procedure, the following error messages may occur; you can ignore them: 3479, 5859, and 11245.

To rebuild master:

1 Use the 11x_bmaster from the ESD#1 distribution to rebuild the master database:

```
11x_bmaster -dmasterdevice_name -sdevice_size -m
```

2 Edit a copy of the *RUN_servername* file in your 12.0 Adaptive Server *install* directory to add the -m flag and the -T3608 trace flag. For example:

```
/sybase/ASE-12_0/bin/dataserver -d/dev/rdsk/c0t2d0s4 \
-sTEST -e/sybase/ASE-12_0/install/TEST_errorlog \
-i/sybase/interfaces -M/sybase/ASE-12_0 -m -T3608
```

Use this file to start the server in single-user mode.

3 For all devices added with disk init, you must run disk reinit. Check the output from sysdevices, if available, for a list of devices. You must give the correct logical and physical names and sizes for the disk reinit command. For more information on disk reinit, see the *Adaptive Server Reference Manual* or the *Adaptive Server Troubleshooting and Error Messages Guide*.

4 Run disk refit. When disk refit completes, Adaptive Server automatically shuts down.

5 Start the server again using the -m flag and the -T3608 trace flag.

6 Run:

```
dbcc checkalloc("master", fix)
```

You can ignore messages similar to the following:

```
EXTID:1536 (Alloc page: 1536) is initialzed. Extent follows:
NEXT=0 PREV=0 OBJID=0 ALLOC=0x0 DEALL=0x0 INDID=1 STATUS=0x0
EXTID:1544 (Alloc page: 1536) is initialzed. Extent follows:
NEXT=0 PREV=0 OBJID=0 ALLOC=0x0 DEALL=0x0 INDID==0 STATUS=0x0
EXTID:1552(Alloc page: 1536) is initialzed. Extent follows:
...
```

7 Shut down the server, and restart the server in the usual way; that is, without the -m flag and the trace flag.

8 Run dbcc checkalloc on all databases. No error messages are expected at this time.

9 Log in to the server, and issue the command:

```
select config_admin(1,122,11920,1,null, null)
```

10 Run the upgrade command from your 12.0 Adaptive Server *upgrade* directory.

11 Run the following commands from your Adaptive Server 12.0 directory:

```
isql -Usa -Ppassword -i scripts/installmaster
isql -Usa -Ppassword -i scripts/instmsgs.ebf
```

If you are using jConnect, run:

```
isql -Usa -Ppassword -i scripts/installjconnect
```

If you are using two-phase commit or distributed transactions, run:

```
isql -Usa -Ppassword -i scripts/installcommit
```

12 Dump all databases.

See Bug #209913.

10.2.2 Rebuilding the model database

To rebuild the model database:

1 Run 11x_bmaster from the ESD#1 distribution:

```
11x_bmaster -dmasterdevice_name -sdevice_size -x
```

2 Edit a copy of the *RUN_servername* file in your 12.0 Adaptive Server *install* directory to add the -m flag. For example:

```
/sybase/ASE-12_0/bin/dataserver -d/dev/rdsk/c0t2d0s4 \
-sTEST -e/sybase/ASE-12_0/install/TEST_errorlog \
-i/sybase/interfaces -M/sybase/ASE-12_0 -m
```

Use this file to start the server in single-user mode.

3 Run:

```
dbcc checkalloc("model", fix)
```

You can ignore messages similar to the following:

```
EXTID:1536 (Alloc page: 1536) is initialzed. Extent follows:
NEXT=0 PREV=0 OBJID=0 ALLOC=0x0 DEALL=0x0 INDID=1 STATUS=0x0
EXTID:1544 (Alloc page: 1536) is initialzed. Extent follows:
NEXT=0 PREV=0 OBJID=0 ALLOC=0x0 DEALL=0x0 INDID==0 STATUS=0x0
EXTID:1552(Alloc page: 1536) is initialzed. Extent follows:
...
```

4 Run:

```
dbcc checkalloc("model")
```

There should be no errors at this time.

5 Run the following command from your 12.0 Adaptive Server directory:

```
isql -Usa -Ppassword < scripts/installmodel
```

6 You can now shut down the server and restart it in your usual way; that is, without the -m flag.

10.3 RPCs from Adaptive Server 12.0 to pre-12.0 versions fail

RPCs from Adaptive Server to earlier server versions fail with a 7221 error, "login failed".

Workaround: Download ESD#1 from the Sybase Web site or from the ESD#1 CD-ROM.

See Bug #209828.

10.4 jConnect exceptions for numeric datatypes

jConnect throws an exception when it attempts to execute a stored procedure with a numeric parameter declared as an output parameter.

Workaround: Download ESD#1 from the Sybase Web site or from the ESD#1 CD-ROM.

See Bug #210686

10.5 *alter table* and triggers with *if update()* clauses

If a trigger contains an if update() clause, data modifications that should fire the trigger that are done after executing alter table add, alter table drop, alter table lock, or alter table modify may cause errors in the column references. Triggers on the altered table that use an if update() clause in the body of the trigger to reference a column may not fire or may fire incorrectly.

Workaround: After the alter table operation has completed, drop and re-create all triggers on the altered table. Doing so causes the if update() clause in the triggers to correctly reference the new columns by their new column offsets, so the trigger code executes correctly.

See Bug #199655.

10.6 Correlated subqueries containing T-SQL outer joins

Adaptive Server version 12.0 does not process correlated subqueries containing Transact-SQL® outer joins the same as earlier versions of Adaptive Server. The following is an example of a query using a correlated variable as the outer member of a Transact-SQL outer join:

```
select t2.b1, (select t2.b2 from t1 where t2.b1
*= t1.a1) from t2
```

Earlier versions of Adaptive Server used trace flag 298 to display error messages for these queries. Depending on whether trace flag 298 was turned on or off and whether the query used an inner or an outer join, Adaptive Server displayed the behavior described in Table 1:

Table 1: Behavior in previous versions of Adaptive Server

Type of query	Trace Flag 298 turned off	Trace Flag 298 turned on
Correlated as an inner member of an outer join	Disallowed: produces error message 11013	No error
Correlated as an outer member of an outer join	No error	Disallowed: produces error message 301

Adaptive Server version 12.0 reverses the behavior of trace flag 298. Because Adaptive Server version 12.0 implements Transact-SQL outer joins as ANSI outer joins during the pre-processor stage, there is an element of risk in allowing such queries to run. Allowing correlated subqueries that contain Transact-SQL outer joins to run with the 298 trace flag turned on is consistent with Sybase's historical trace flag usage. For version 12.0, the behavior of trace flag 298 is:

Table 2: Behavior in Adaptive Server version 12.0

Type of query	Trace flag 298 turned off	Trace flag 298 turned on
Correlated as an inner member of an outer join	Disallowed: produces error message 11013	Disallowed: produces error message 11013
Correlated as an outer member of an outer join	Disallowed: produces error message 11055	No error

Adaptive Server version 12.0 changes error message 301 to error message 11055, although the text of the message remains the same.

See Bug #194304.

10.7 Cursor fetch using CIS to query a proxy table

When using Component Integration Services (CIS) in Adaptive Server 12.0 to query a proxy table, which is mapped to server class DirectConnect, a cursor fetch fails if a user transaction is not started before the cursor is declared.

Workaround: Begin a transaction before declaring the cursor.

See Bug #196786.

10.8 Error when viewing table properties in Sybase Central

When you select a user table and click Properties, then click the Miscellaneous tab, an error message appears. If you click OK, the error message dialog closes; however, if you then click the properties button on the top of the Miscellaneous sheet, next to the drop-down labelled Segment, Sybase Central causes an application error and is shutdown.

A similar error message also appears when you connect to a data server, select the *execution classes* folder, view the properties for one of the execution classes shown, then select the bindings tab. An error dialog appears with a message similar to the one in the previous scenario; however, in this case Sybase Central does not cause an application error.

Workaround: To query the server for the table information, use isql:

```
use db_name
go
SELECT rowcnt( doampg ),
    reserved_pgs( id, doampg ) + reserved_pgs( id, ioampg ),
    data_pgs( id, doampg ), data_pgs( id, ioampg )
    FROM sysindexes WHERE id = id
go
```

See Bug #209782.

10.8.1 Highlighted known problems with Sybase Failover

The following sections describe known problems and workarounds for Adaptive Servers configured with Sybase Failover.

Logins not locked across companions

If a user attempts to connect to a companion server, and the number of attempts fail more times than sp_configure max failed logins allows, the user's login is locked in the local server, but the remote companion does not lock this user's login. For example, if primary companion has a max failed logins set to 4, and user alison attempts to connect but enters her password wrong 5 times in a row, her account is locked on the local companion. However, she could still log in to the secondary companion.

Workaround: Issue sp_locklogin in the secondary companion to prevent user alison from logging in to the secondary companion.

See Bug #199463.

number of retries and *loopdelay* values cause delayed client failover

If you specify a large number for either the number of retries or the loopdelay values in your *interfaces* file, clients configured with the failover property (for example, isql -Q) can take a very long time to fail over from the primary companion to the secondary companion. For example, if you specify a retry of 10 and a loop delay of 240, the client tries to reconnect to the failed companion server every 240 seconds for ten times before eventually failing over.

Workaround: Specify a low value for number of retries. However, be aware that specifying a value of zero can cause spurious failover attempts because of network problems.

See Bug #206745.

Erroneous *local* server entry added by *srvbuild*

When you create an Adaptive Server, srvbuild adds a server entry named 'local' (the name of the server with *srvid*>0) as one of the remote servers in the sysservers table in the master database.

During configuration, if the erroneous 'local' server entries have different server IDs in the companion servers, sp_companion tries to adjust for compatible server IDs. However, while adjusting, sp_companion incorrectly tries to add the erroneous 'local' server entry as a real local server (*srvid*=0), which already exists, so the configuration is aborted. If srvbuild is used to generate the master device, it usually uses the same server IDs, so you may not see this problem.

Workarounds: Before issuing the configuration command, make sure that either:

- Server entries named 'local' that are *not* the real local servers (*srvid*=0) have the same server IDs as the real local servers, or
- The local server entries are dropped in both servers.

See Bug #209524.

create schema causes lock contention

The following situation applies only if you have configured the companion servers with sp_companion...with_proxydb.

When you issue create schema on a companion server, the schema replication to the proxy database causes lock contention. create schema is not supported in normal companion mode and causes the companions to hang. This problem does not affect the other create commands.

See Bug #197796.

Do not use *disk init* to create device in failed-over file system

When a companion server is in failover mode, *never* use disk init to create a local device in the file system of the failed-over companion, even though this command allows you to. disk init does not issue a warning message. Before you create a new device, make sure your companions are not in failover mode.

If you create a local device on the failed-over file system, the secondary companion is killed after you run sp_companion...prepare_failback, and you have to manually restart the secondary companion. (The steps for performing this action are platform-dependent. For details, see the configuration chapter for your platform in *Using Sybase Failover in a High Availability System*.) When you restart the secondary companion, it cannot recover any databases you created on the local devices; it marks them suspect. Also, the device you created on the secondary companion's file system is marked offline, and cannot be dropped.

To recover from this situation, perform the same steps as to recover from a database marked suspect.

See Bug #206000.

dbcc checkstorage cannot read from mounted *dbcc* databases

Before you run dbcc checkstorage on any failed-over system databases, you must configure the dbcc_config table in the local dbcc databases (dbccdb and dbccalt) with the appropriate configuration information. Because of the following reasons, dbcc checkstorage cannot read configuration information from mounted dbcc databases (or store results in mounted dbcc databases):

- *Unavailable dbids* – during failover, the dbids for user databases remain the same, but the dbids for system databases change. After failover, the dbcc_config table in the local dbcc database will not automatically have rows for checking any of the failed-over system databases. For example, if companion server S1 fails over to companion server S2, the dbcc_config table on S2 does not contain the dbids for the system databases after failover.

- *Incorrect configuration information* – if the configuration information for checking failed-over user databases is read from dbcc_config in the mounted dbcc database where the information is available (companion server S2 in the scenario above), this information may not be valid for the server on which databases are now mounted. For example, the secondary companion may not be configured with the same named caches as the primary server that failed over. dbcc checkstorage aborts with an error if the dbcc_config entries in the mounted dbcc database are not properly configured or are missing rows. Also, you cannot alter mounted dbcc databases if there is not space for storing results.
- *Unavailable checkstorage results after failback* – after failback, any dbcc checkstorage results generated during failover mode for mounted databases are not available in the primary dbcc databases. These results are stored in the dbcc_operation_results table in the dbcc database local to the secondary server (companion server S2 in the example above) instead of in the mounted dbcc database.

See Bug #202062.

Change permissions of the *ha_companion* file

When you run sp_companion, *primary_companion_name* configure, a file called *ha_companion* is created under */var/opt/sybase* of the secondary companion. You must manually change the permissions of this file so that it is readable and writable only by the "sybase" user.

See Bug #209047.

11. Documentation updates and clarifications

This section describes changes to the following Adaptive Server documents:

- Installation Guide
- What's New in Adaptive Server Enterprise 12.0
- Configuration Guide for Windows 95, 98, and NT
- Transact-SQL User's Guide
- System Administration Guide
- Adaptive Server Reference Manual
- Java in Adaptive Server Enterprise

- Using Sybase Failover in a High Availability System
- Component Integration Services User's Guide
- Online Help for ASE plug-in to Sybase Central

Note "Sybooks-on-the-Web" is the obsolete name for the "Technical Library."

11.1 Installation Guide

This section provides corrections to the Adaptive Server 12.0 *Installation Guide*.

11.1.1 Incorrect paragraph in section about unloading products

In the Adaptive Server 12.0 *Installation Guide*, Chapter 3, "Unloading Server Products from Distribution Media," in the section "Unloading Components with Studio Installer: GUI Mode," ignore the following paragraph (sixth down from the beginning of the section):

"During the installation process, the computer must be restarted. After restarting, the computer accesses the CD for additional data. For this reason, the CD must remain in the CD-ROM drive during restart, and the CD-ROM drive should be on the installation machine."

This information does not apply to UNIX platforms.

11.1.2 Using *dbcc upgrade_object*

The Adaptive Server 12.0 *Installation Guide* did not mention upgrade issues related to using dbcc upgrade_object. See "dbcc upgrade_object issues" on page 9 for more information and instructions.

11.2 What's New in Adaptive Server Enterprise 12.0

The following sections describe changes to *What's New in Adaptive Server Enterprise 12.0?*

11.2.1 Correction to “Disabling Triggers” section

In Chapter 1, “New Features in Adaptive Server Version 12,” the text in the section “Disabling Triggers” is incorrect. Replace the section with the following:

bcp and insert fire any triggers they encounter while they copy data into a table, which increases the amount of time it takes to perform the copy (for example, bcp switches from fast mode to slow mode to fire the trigger). Use the disable trigger option of the alter table command to disable any triggers associated with a table before you copy in the data. You can use the disable trigger option to either disable all the triggers associated with the table, or you can specify a particular trigger to disable. However, any triggers you disable will not be fired after the copy is complete. If you require these triggers to insert, update, or delete any data, you will have to fire them manually.

alter table... disable trigger uses the following syntax:

```
alter table [database_name.[owner_name].]table_name
{enable | disable} trigger [trigger_name]
```

11.2.2 Changes to table-level locking in version 11.9.2

The following section is an addition to Chapter 3, “New Features in Adaptive Server Version 11.9.2.”

In version 11.9.2, performing an update or delete via a table scan does not acquire a table-level exclusive lock as it did in earlier versions. In version 11.9.2, the scan acquires an exclusive intent table lock when the transaction starts. When the scan locates a row that must be updated, it acquires an exclusive page or row lock, depending on the locking scheme. The only exception is an unindexed update or delete to a data-only-locked table at transaction isolation level 3; these updates acquire an exclusive table lock. In previous versions, any update or delete that does not use an index acquires a table-level exclusive lock at the start of the transaction.

In most cases, performing the scan without using a table level lock increases concurrency. However, some applications may experience one of these problems:

- Applications that did not deadlock in earlier versions may experience deadlocks in 11.9.2.
- Tasks that update a large number of rows may use a large number of locks if they cannot perform lock promotion due to conflicting locks.

Workarounds:

- If deadlocks are creating problems, consider converting the table to use datarows locking.
- If deadlocks or excessive numbers of locks are creating problems, consider using the lock table command to acquire an exclusive table lock before you begin the transaction.

11.3 Configuration Guide for Windows 95, 98, and NT

The following sections describe changes to the *Configuration Guide for Windows 95, Windows 98, and Windows NT*.

11.3.1 Update to Adaptive Server platforms statement

The first paragraph in Chapter 1, "Overview of Adaptive Server Enterprise," is not correct. The correct text is:

Adaptive Server Enterprise for Windows NT is a full-featured Adaptive Server that runs on the Windows NT operating system. Adaptive Server does not run on Windows 95 or Windows 98. However, the PC-client products that are delivered with Adaptive Server run on Windows 95, Windows 98 and Windows NT.

11.3.2 Update to connection information formats for IPX/SPX

In Chapter 3, "Setting Up Communications Across the Network," in the section "Available NWLink IPX/SPX Connection Formats," Table 3-1: "Connection information formats for IPX/SPX" describes three formats available for NWLink IPX/SPX MASTER and QUERY entries. For Adaptive Server 12.0 only format 1, *net_number*, *node_number*, *socket_number*, is supported.

11.3.3 Update to procedure for increasing the priority of Adaptive Server

In Chapter 8, "Managing Adaptive Server Databases," in the section, "Using Dedicated Adaptive Server Operation," the information about setting the default NT tasking option to give the foreground application, Adaptive Server, the best application response time is incorrect.

Because Adaptive Server 12.0 runs as an NT service, it will not be the foreground application. Following the steps to set the default NT tasking option shown in the Configuration Guide reduces the CPU time available for Adaptive Server. Instead, use the following procedure.

To increase the priority of Adaptive Server:

1 Start the Server Config tool either from the Sybase menu or from the Sybase Central Utilities folder.

2 Select the Configure Adaptive Server option.

3 Select the server to configure, then click Continue.

4 Respond Yes if the server needs to be started, and enter an "sa" login and password when prompted.

5 Select Command Line Parameters.

6 Enter -P in the parameter entry field.

7 Click OK.

When the server restarts, it picks up this new command-line parameter.

11.3.4 Update to procedure for configuring the ODBC driver

In Chapter 3, "Setting Up Communications Across the Network," the procedure in the section "Configuring the Driver" is not correct. Replace that procedure with the following paragraphs:

Before you can use a driver, you must configure a data source for the driver and set up an Open Client connection using dsedit. A data source consists of a data source name, driver location, and optional driver information in the system information.

Use the ODBC Administrator to select an installed driver, and then configure a data source for it. (If you do not have the latest Microsoft Data Access Components installed, you receive an error message.) Configuration instructions are provided in the online help for each driver's configuration window.

11.4 Transact-SQL User's Guide

The following sections discuss changes to the *Transact-SQL User's Guide*.

11.4.1 Updates to "Delimited Identifiers" section

In Chapter 1, "Introduction," in the section "Delimited Identifiers" add the following statement:

A pound sign (#) is not legal as a first character of any quoted identifier.

This restriction applies to Adaptive Server 11.5 and all later versions.

11.4.2 Correction to “Disabling Triggers” section

In Chapter 16, “Triggers: Enforcing Referential Integrity,” in the section “Disabling Triggers” the first paragraph is incorrect. Replace the paragraph with the following:

“bcp and insert fire any triggers they encounter while they copy data into a table, which increases the amount of time it takes to perform the copy (for example, bcp switches from fast mode to slow mode to fire the trigger). Use the disable trigger option of the alter table command to disable any triggers associated with a table before you copy in the data. You can use the disable trigger option to either disable all the triggers associated with the table, or you can specify a particular trigger to disable. However, any triggers you disable will not be fired after the copy is complete. If you require these triggers to insert, update, or delete any data, you will have to fire them manually.”

11.5 System Administration Guide

The following section describes changes to the *System Administration Guide*.

11.5.1 Updates to Chapter 17, “Setting Configuration Parameters”

The following configuration parameters were not included:

- check password for digit
- maximum failed logins
- minimum password length

They are described in the following sections.

check password for digit

Use the check password for digit configuration parameter to check for at least one character or digit in a password. Setting this parameter does not affect existing passwords. By default checking for digits is off. This parameter is server-wide.

For example:

```
sp_configure "check password for digit", 1
```

Activates the check password functionality.

This dynamic configuration parameter applies on a server-wide basis. It can be set by the System Security Officer only. This parameter is a toggle, 1 for on, 0 for off. The default value is 0.

maximum failed logins

Use the maximum failed logins configuration parameter to set the server-wide maximum number of login attempts for logins and roles.

For example:

```
sp_configure "maximum failed logins", 5
```

Sets the server-wide maximum number of failed login attempts to 5.

This dynamic configuration parameter applies on a server-wide basis. It can be set by the System Security Officer only. It can take any value between 0 and 32767. The default value is 0.

minimum password length

Use the minimum password length configuration parameter to specify a server-wide value for minimum password length for both logins and roles.

For example:

```
sp_configure "minimum password length", 4
```

Sets the minimum password length to four characters.

This dynamic configuration parameter applies on a server-wide basis. It can be set by the System Security Officer only. It can take any value between 0 and 30 inclusive. The default value is 6.

11.5.2 Using *sp_sendmsg* or the *syb_sendmsg* function

The syb_sendmsg function sends a message up to 255 bytes in size to another application from Adaptive Server to User Datagram Protocol (UDP) port. The arguments to syb_sendmsg are the IP address and port number on the destination host and the message to be sent. You can use either the syb_sendmsg function, or use sp_sendmsg, which includes the function in a system procedure.

Note syb_sendmsg and sp_sendmsg are not supported on Windows NT.

The UDP protocol does not return an acknowledgment, so Adaptive Server does not guarantee the receipt of the message by the receiving application. The port value can be any port not already in use by another process, excluding ports 1– 1024, up to 65535 or the system limit.

This example uses syb_sendmsg to send the message “Hello” to port 3456 at IP address ‘120.10.20.5’:

```
select syb_sendmsg("120.10.20.5", 3456, "Hello")
```

This example uses sp_sendmsg to do the same thing:

```
sp_sendmsg "120.10.20.5", 3456, "Hello"
```

This example uses syb_sendmsg, reading the IP address and port number from a table, using a variable for the message:

```
declare @msg varchar(255)
select @msg = "Message to send"
select syb_sendmsg (ip_address, portnum, @msg)
from sendports
where username = user_name()
```

You can also specify the IP address and port numbers as variables.

Note There are no security checks with syb_sendmsg. Sybase strongly recommends that you not use syb_sendmsg to send sensitive information across the network. By enabling this functionality, the user accepts any security problems which result from its use.

To send messages, a System Security Officer must enable the syb_sendmsg using sp_configure:

```
sp_configure "allow sendmsg", 1
```

The allow sendmsg parameter is dynamic.

To specify the port number that Adaptive Server uses for the outgoing messages, use:

```
sp_configure "syb_sendmsg port number", port_number
```

If multiple engines are configured, one port number is used for each engine, numbered consecutively.

The syb_sendmsg *port number* parameter requires a restart of the server. It can be set by a System Administrator.

Sample C program

This sample C program listens on a UDP port that you specify and performs some error checking. For example, to receive the syb_sendmsg calls for the example above, use:

```
updmon 3456

#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <fcntl.h>

main(argc, argv)
int argc; char *argv[];
{
     struct sockaddr_in sadr;
     int portnum,sck,dummy,msglen;
     char msg[256];

     if (argc < 2) {
          printf("Usage: udpmon <udp portnum>\n");
          exit(1);
     }

     if ((portnum=atoi(argv[1])) < 1) {
          printf("Invalid udp portnum\n");
          exit(1);
     }

     if ((sck=socket(AF_INET,SOCK_DGRAM,IPPROTO_UDP)) < 0) {
          printf("Couldn't create socket\n");
          exit(1);
     }

     sadr.sin_family = AF_INET;
     sadr.sin_addr.s_addr = inet_addr("0.0.0.0");
     sadr.sin_port = portnum;

     if (bind(sck,&sadr,sizeof(sadr)) < 0) {
          printf("Couldn't bind requested udp port\n");
          exit(1);
     }
```

```
     for (;;)
          {
               if((msglen=recvfrom(sck,msg,sizeof(msg),0,NULL,&dummy)) < 0)
                    printf("Couldn't recvfrom() from udp port\n");
               printf("%.*s\n", msglen, msg);
          }
}
```

11.6 Adaptive Server Reference Manual

The following sections describe changes to the *Adaptive Server Reference Manual*.

11.6.1 Updates to *compare* and *sortkey*

Add the following information to the syntax description for the compare and sortkey functions.

There are two types of collation tables, built-in and external. You can use either the collation name or the collation ID to specify a built-in table. You must use the collation name to specify external tables. Table 3 lists the valid values for *collation_name* and *collation_ID*.

Table 3: Collation names and IDs

Description	Collation name	Collation ID
Binary sort	binary	50
Default Unicode multilingual	default	0
CP 850 Alternate: no accent	altnoacc	39
CP 850 Alternate: lower case first	altdict	45
CP 850 Alternative: no case preference	altnocsp	46
CP 850 Scandinavian dictionary	scandict	47
CP 850 Scandinavian no case preference	scannocp	48
GB Pinyin	gbpinyin	n/a
Latin-1 English, French, German dictionary	dict	51
Latin-1 English, French, German no case	nocase	52
Latin-1 English, French, German no case preference	nocasep	53
Latin-1 English, French, German no accent	noaccent	54
Latin-1 Spanish dictionary	espdict	55
Latin-1 Spanish no case	espnocs	56
Latin-1 Spanish no accent	espnoac	57

Description	Collation name	Collation ID
ISO 8859-5 Cyrillic dictionary	cyrdict	n/a
ISO 8859-5 Russian dictionary	rusdict	n/a
ISO 8859-9 Turkish dictionary	turdict	n/a
Shift-JIS binary order	sjisbin	n/a
Thai dictionary	thaidict	1

11.6.2 Update to *license_enabled*

The permissions section of manual page for the license_enabled function is incorrect. It should read:

Any user can execute license_enabled.

11.6.3 Update to *alter table*

The following paragraph on page 6-34 of the *Commands Reference Manual* needs to be updated:

"You cannot drop columns that contain or are referenced by rules, constraints, or defaults. Instead, first drop the rule, constraint, or default, then drop the column. Use sp_helpconstraint to identify any constraints on a table, and use sp_depends to identify any column- level dependencies."

It should state:

"You can drop columns that have defaults or rules bound to them. Any column-specific defaults are also dropped when you drop the column. You cannot drop columns that have check constraints or referential constraints bound to them. Instead, first drop the check constraint or referential constraint, then drop the column. Use sp_helpconstraint to identify any constraints on a table, and use sp_depends to identify any column-level dependencies.

11.6.4 Update to *create table*

The information about identity_gap was not included in the create table manual page. The correct syntax is:

create table *table_name* (*column_name datatype*(*constant_expression*) identity)
with identity_gap = *value*

For example:

```
create table mytable (IdNum numeric(12,0) identity)
with identity_gap = 10
```

This statement creates a table named mytable with an identity column. The identity gap is set to 10, which means ID numbers will be allocated in memory in blocks of ten. If the server fails or is shut down with no wait, the maximum gap between the last ID number assigned to a row and the next ID number assigned to a row is ten numbers.

For more information about identity gaps, see the section "Managing Identity Gaps in Tables" in Chapter 7, "Creating Databases and Tables" in the *Transact-SQL User's Guide*.

11.6.5 Updates to *disk init* and *disk reinit*

The dsync parameter was not included in the manual pages for disk init and disk reinit. The following sections describe the parameter for both commands.

Function

The dsync keyword defines the dsync setting for new database device files.

Syntax

```
disk init
    name = "device_name" ,
    physname = "physicalname" ,
    vdevno = virtual_device_number ,
    size = number_of_blocks
    [, vstart = virtual_address ,
    cntrltype = controller_number ]
   [, dsync = {true | false} ]

disk reinit
    name = "device_name" ,
    physname = "physicalname" ,
    vdevno = virtual_device_number ,
    size = number_of_blocks
    [, vstart = virtual_address ,
    cntrltype = controller_number ]
   [, dsync = {true | false} ]
```

Keyword

dsync – specifies whether writes to the database device take place directly to the storage media, or are buffered when using UNIX operating system files. This option is meaningful only when you are initializing a UNIX operating system file; it has no effect when initializing devices on a raw partition. By default, all UNIX operating system files are initialized with dsync set to true.

Examples

```
disk init
name = "user_file",
physname = "/usr/u/sybase/data/userfile1.dat",
vdevno = 2, size = 5120, dsync = true
```

Initializes 10MB of a disk on a UNIX operating system file. Adaptive Server opens the device file with the dsync setting, and writes to the file are guaranteed to take place directly on the storage media. Use dsync the same way with disk reinit.

Comments

- When dsync is true, writes to the database device are guaranteed to take place on the physical storage media, and Adaptive Server can recover data on the device in the event of a system failure.
- When dsync is false, writes to the database device may be buffered by the UNIX file system. The UNIX file system may mark an update as being completed, even though the physical media has not yet been modified. In the event of a system failure, there is no guarantee that data updates have ever taken place on the physical media, and Adaptive Server may be unable to recover the database.

  Note Do not set dsync to false for any device that stores critical data. The only exception is tempdb, which can safely be stored on devices for which dsync is set to false.

- dsync is always true for the master device file.
- dsync should be false only when databases on the device need not be recovered after a system failure. For example, you may consider setting dsync to false for a device that stores only the tempdb database.
- Adaptive Server ignores the dsync setting for devices stored on raw partitions—writes to those device are guaranteed to take place on the physical storage media, regardless of the dsync setting.
- The dsync setting is not used on the Windows NT platform.
- disk reinit ensures that master..sysdevices is correct if the master database has been damaged or if devices have been added since the last dump of master.

11.6.6 Update to *load database file=filename* description

When using load database, the dumpvolume option does not provide an error messages if an incorrect file name is given for the file=*filename* option. The Backup Server searches the entire tape looking for that file, regardless of an incorrect tape mounted.

11.6.7 Correction to *reorg rebuild syntax*

The syntax for reorg rebuild in the manual page is incorrect. The correct syntax is:

reorg rebuild *tablename* [*indexname*]

You must supply the table name. Optionally, you can add the index name.

11.6.8 Update to *select into*

The information about the identity_gap parameter was not included in the select manual page. This parameter works the same way in a select into statement as it does in create table.

If you are creating a table in a select into statement from a table that has a specific identity gap setting, the new table does not inherit the identity gap setting from the parent table. Instead, the new table uses the identity burning set factor setting. To give the new table a specific identity_gap setting, specify the identity gap in the select into statement. You can give the new table an identity gap that is the same as or different from the parent table.

For example, to create a new table (newtable) from the existing table (oldtable) with an identity gap, you specify it in the select into statement:

```
select identity into newtable
with identity_gap = 20
from oldtable
```

For more information about identity gaps, see the section "Managing Identity Gaps in Tables" in Chapter 7, "Creating Databases and Tables" in the *Transact-SQL User's Guide*.

11.6.9 Update to *set*

The set transaction isolation command can accept a variable, for example:

```
select @level = 0
set transaction isolation level @level
go
```

However, this syntax should not be used within compiled objects, for example, such as procedures or triggers, because the compiler needs to know what transaction isolation level will be in effect. If you use a variable in a compiled object, unexpected and unpredictable results may be found.

11.6.10 Update to *sp_chgattribute*

The information about identity_gap was not included in the sp_chgattribute manual page. The correct syntax is:

sp_chgattribute "*table_name*", "identity_gap", *set_number*

where:

- *table_name* is the name of the table for which you want to change the identity gap.
- identity_gap indicates that you want to change the identity gap.
- *set_number* is the new size of the identity gap.

For example:

```
sp_chgattribute "mytable", "identity_gap", 20
```

To change mytable to use the identity burning set factor setting instead of the identity_gap setting, set identity_gap to 0:

```
sp_chgattribute "mytable", "identity_gap", 0
```

For more information about identity gaps, see the section "Managing Identity Gaps in Tables" in Chapter 7, "Creating Databases and Tables" in the *Transact-SQL User's Guide*.

11.6.11 Correction to *sp_dbcc_summaryreport*

In Chapter 10, "dbcc Stored Procedures" the syntax for sp_dbcc_summaryreport is incorrect. The correct syntax is:

sp_dbcc_summaryreport [*dbname* [, *date*] [, *opname*]]

where:

dbname specifies the name of the database for which you want the report generated. If you do not specify *dbname*, sp_dbcc_summaryreport generates reports on all databases in dbccdb..dbcc_operations_log for which the date is on or before the date and time specified by the *date* option.

date specifies the date on which dbcc checkstorage was performed. If you do not specify a date, sp_dbcc_summaryreport uses the date of the last dbcc checkstorage operation performed on the target database. This parameter is of the datatype datetime. If both the date and the time are specified for *date*, summary results of all the operations performed on or before the specified time are reported. If no dated is specified, all operations are reported.

opname specifies the operation. *opname* may be checkstorage, which is the default, or checkverify, or both. If *opname* is not specified, reports are generated for all operations.

11.6.12 Updates to *sp_depends*

The following item should be added to the "Comments" section of the sp_depends manual page:

- Before you modify or drop a column, use sp_depends to determine if the table contains any dependent objects that could be affected by the modification. For example, if you modify a column to use a new datatype, objects tied to the table may need to be redefined to be consistent with the column's new datatype.

Examples

The sp_depends manual page should contain the following examples:

1 The following example shows the column-level dependencies for all columns of the sysobjects table:

```
sp_depends sysobjects

Things inside the current database that reference the object.
object                                  type
--------------------------------------- ----------------
dbo.sp_dbupgrade                        stored procedure
dbo.sp_procxmode                        stored procedure

Dependent objects that reference all columns in the table. Use sp_depends
on each column to get more information.
Columns referenced in stored procedures, views or triggers are not
included in this report.

Column           Type         Object Names or Column Names
---------------------- ------------ ----------------------------
cache            permission   column permission
ckfirst          permission   column permission
crdate           permission   column permission
deltrig          permission   column permission
expdate          permission   column permission
id               index        sysobjects (id)
id               logical RI   From syscolumns (id) To sysobjects (id)
id               logical RI   From syscomments (id) To sysobjects (id)
id               logical RI   From sysdepends (id) To sysobjects (id)
id               logical RI   From sysindexes (id) To sysobjects (id)
id               logical RI   From syskeys (depid) To sysobjects (id)
id               logical RI   From syskeys (id) To sysobjects (id)
```

```
id              logical RI   From sysobjects (id) To sysprocedures (id)
id              logical RI    From sysobjects (id) To sysprotects (id)
id              logical RI   sysobjects (id)
id              permission   column permission
indexdel        permission   column permission
instrig         permission   column permission
loginame        permission   column permission
name            index        ncsysobjects (name, uid)
name            permission   column permission
objspare        permission   column permission
schemacnt       permission   column permission
seltrig         permission   column permission
sysstat         permission   column permission
sysstat2        permission   column permission
type            permission   column permission
uid             index        ncsysobjects (name, uid)
uid             logical RI   From sysobjects (uid) To sysusers (uid)
uid             permission   column permission
updtrig         permission   column permission
userstat        permission   column permission
versionts       permission   column permission
```

2 The following example shows more details about the column-level dependencies for the id column of the sysobjects table:

```
sp_depends sysobjects, id

Things inside the current database that reference the object.
object                                          type
-----------------------------------             -------------
dbo.sp_dbupgrade                                stored procedure
dbo.sp_procxmode                                stored procedure
Dependent objects that reference column id.
Columns referenced in stored procedures, views or triggers are not
included in this report.
Type          Property     Object Names or Column Names
                           Also see/Use command
----------   ---------     --------------------------------
                           --------------------------------
index         index        sysobjects (id)
                           sp_helpindex, drop index,
                           sp_helpconstraint, alter table drop constraint
logical RI    primary      sysobjects (id)
                           sp_helpkey, sp_dropkey
logical RI    foreign      From syskeys (id) To sysobjects (id)
                           sp_helpkey, sp_dropkey
```

```
logical RI    common      From syscolumns (id) To sysobjects (id)
                          sp_helpkey, sp_dropkey
logical RI    common      From sysdepends (id) To sysobjects (id)
                          sp_helpkey, sp_dropkey
logical RI    common      From sysindexes (id) To sysobjects (id)
                          sp_helpkey, sp_dropkey
logical RI    common      From syskeys (depid) To sysobjects (id)
                          sp_helpkey, sp_dropkey
logical RI    common      From syscomments (id) To sysobjects (id)
                          sp_helpkey, sp_dropkey
logical RI    common      From sysobjects (id) To sysprotects (id)
                          sp_helpkey, sp_dropkey
logical RI    common      From sysobjects (id) To sysprocedures (id)
                          sp_helpkey, sp_dropkey
permission    permission  column permission
                          sp_helprotect, grant/revoke
```

3 The following example shows the column-level dependencies for all columns of the user-created table, titles:

```
1> sp_depends titles

Things inside the current database that reference the object.
object                                  type
----------------------------------  ---------------
dbo.deltitle                            trigger
dbo.history_proc                        stored procedure
dbo.title_proc                          stored procedure
dbo.titleid_proc                        stored procedure
dbo.titleview                           view
dbo.totalsales_trig                     trigger

Dependent objects that reference all columns in the table. Use sp_depends
on each column to get more information.
Columns referenced in stored procedures, views or triggers are not
included in this report.

Column     Type         Object Names or Column Names
------     -----        ----------------------------------------
pub_id     logical RI   From titles (pub_id) To publishers (pub_id)
pubdate    default      datedflt
title      index        titleind (title)
title      statistics   (title)
title_id   index        titleidind (title_id)
title_id   logical RI   From roysched (title_id) To titles (title_id)
title_id   logical RI   From salesdetail (title_id) To titles (title_id)
title_id   logical RI   From titleauthor (title_id) To titles (title_id)
title_id   logical RI   titles (title_id)
```

```
title_id   rule         title_idrule
title_id   statistics   (title_id)
type       default      typedflt
```

4 The following example shows more details about the column-level dependencies for the pub_id column of the user-created titles table:

```
sp_depends titles, pub_id

Things inside the current database that reference the object.
object                                  type
-----------------------------------     ----------------
dbo.deltitle                            trigger
dbo.history_proc                        stored procedure
dbo.title_proc                          stored procedure
dbo.titleid_proc                        stored procedure
dbo.titleview                           view
dbo.totalsales_trig                     trigger
Dependent objects that reference column pub_id.
Columns referenced in stored procedures, views or triggers are not
included in this report.
Type          Property    Object Names or Column Names
                          Also see/Use command
----------    ---------   --------------------
                          ----------------------------
logical RI    foreign     From titles (pub_id) To publishers (pub_id)
                          sp_helpkey, sp_dropkey
```

11.6.13 Update to *sp_displaylogin*

sp_displaylogin displays the login security-related parameters configured for a login. The syntax has not changed; however, the output shows the following additional information:

- Whether the account is locked
- The date of the last password change
- The password expiration interval
- Whether the password has expired
- The minimum length of the password
- The maximum number of failed logins allowed before the login is locked
- The current number of failed logins

Example

```
sp_displaylogin joe

Suid: 294
Loginame: joe
Fullname: Joseph Resu
Default Database: master
Default Language:
Configured Authorization: intern_role (default OFF)
Locked: NO
Date of Last Password Change: Nov 24 1998 3:46PM
Password expiration interval : 5
Password expired : NO
Minimum password length:4
Maximum failed logins : 10
Current failed logins : 3
```

11.6.14 Update to *sp_displayroles*

sp_displayroles displays the login security-related parameters configured for a role. The information about the display_info option was not included in the sp_displayroles manual page.

Syntax

sp_displayroles [*grantee_name* [, *mode*]]

where:

grantee_name is the name of the role or login for which you want to display the security related configuration.

mode is one of the following:

- expand_up, which shows the role hierarchy tree for the parent levels
- expand_down, which shows the role hierarchy tree for the child levels
- display_info, which shows the login security-related parameters configured for the specified role

The output of sp_displayroles shows the following additional information:

- Whether the account is locked
- The date of the last password change
- The password expiration interval
- Whether the password has expired
- The minimum length of the password
- The maximum number of failed logins allowed before the login is locked

- The current number of failed logins

Example

```
sp_displayroles physician_role, "display_info"
Role name = physician_role
Locked : NO
Date of Last Password Change : Oct 31 1999 3:33PM
Password expiration interval = 5
Password expired : NO
Minimum password length = 4
Maximum failed logins = 10
Current failed logins = 3
```

For more information, see “User-Defined Login Security” in Chapter 5, “Security Administration,” in the *System Administration Guide*.

11.6.15 Correction to *sp_helprotect* example

In the sp_helprotect manual page, example 5 reads:

```
sp_helprotect @role_name = doctor_role
```

The syntax for sp_helprotect does not allow @role_name to be specified without the other parameters being specified. The example should read:

```
sp_helprotect doctor_role
```

11.6.16 Updates to *sp_modifylogin*

Use sp_modifylogin to change password expiration interval, the minimum password length, and the maximum number of failed logins allowed for a specified existing login.

Syntax

sp_modifylogin {*loginame* | "all overrides"} , "*option*", "*value*"

where:

loginame is the login account to be modified.

"all overrides" removes the system overrides that were set using the “passwd expiration”, "min passwd length", or "max failed_logins" parameters. To remove all the login-specific values, specify:

```
"all overrides" "column" -1
```

option is one of the following values:

Option	Definition
"passwd expiration"	The password expiration interval in days. It can be any value between 0 and 32767, inclusive.

Option	Definition
"min passwd length"	The minimum password length required for the specified login. It can be any value between 0 and 30, inclusive. 0 specifies that no password is required. The default is 6.
"max failed_logins"	The number of allowable failed login attempts for the specified login. It can be any value between 0 and 32767, inclusive.

The *option* parameter also includes the following values that were available for the *column* parameter in versions prior to 12.0:

Option	Definition
defdb	The "home" database to which the user is connected when he or she logs in.
deflanguage	The official name of the user's default language.
fullname	The user's full name.
"add default role"	The role or roles to be activated by default at login.
"drop default role"	The role or roles to be dropped from the list of roles activated by default at login.

Note The *column* parameter is not valid in Adaptive Server 12.0. If you specified the *column* parameter name in your implementation of sp_modifylogin in an earlier version, you must replace it with *option*.

value is the value of the option you specified for the *option* parameter. The *value* parameter is a character datatype; therefore, quotes are required for positive and negative numeric values.

Examples

```
sp_modifylogin "joe", @option="max failed_logins",
@value="40"
```

Changes the maximum number of failed login attempts for the login "joe" to 40.

Note The value parameter is a character datatype; therefore, quotes are required for numeric values.

```
sp_modifylogin "all overrides", "max failed_logins",
"3"
```

Changes the overrides for maximum failed login attempts of all logins to 3.

```
sp_modifylogin "all overrides", @option="max
failed_logins", @value="-1"
```

Removes the overrides for maximum failed logins option for all logins.

For more information about password expiration interval, minimum password length, and maximum number of failed logins, see "User-Defined Login Security" in Chapter 5, "Security Administration," in the *System Administration Guide*.

11.6.17 *sp_syntax* not supported

sp_syntax is not supported in Adaptive Server 12.0. Disregard any mention of this system procedure.

11.6.18 Update to auditing events table

In the sysaudits_01 – sysaudits_08 manual page, Table 11-13 shows the values in the event and extra info columns. Add the following row to this table:

Audit option	event	Command or access audited	extrainfo
cmdtxt	92	All actions of a particular user, or by users with a particular role	*Roles:* Current active roles *Subcommand:* NULL *Previous value:* NULL *Current value:* NULL *Other information:* NULL *Proxy information:* Original login name if a set proxy is in effect

11.7 Java in Adaptive Server Enterprise

This section describes changes to *Java in Adaptive Server Enterprise*.

11.7.1 Do not send or receive Java objects through an RPC

Java objects should not be sent as parameters to an RPC or received from an RPC. They do not translate correctly.

11.8 Using Sybase Failover in a High Availability System

In addition to the manual, *Using Sybase Failover in a High Availability System*, see the technical note/white paper, *Configuring Sybase ASE 12.0 for High Availability on Sun Cluster HA* at http://techinfo.sybase.com/css/techinfo.nsf/White+Papers.

11.8.1 You must install *hasybase* agent

If you are configuring Adaptive Server as a companion server in a high availability system, you must install Sun's hasybase agent before you install Adaptive Server.

11.9 Component Integration Services User's Guide

The following sections describe changes to the *Component Integration Services User's Guide*.

11.9.1 Updated example of remote table definition

In Chapter 2, "Understanding Component Integration Services," in the section "Topics and Issues," replace the section "Example of Remote Table Definition" with the following section.

Example of remote table definition

The following example illustrates the steps necessary to define the remote Adaptive Server table, authors, starting with the server definition:

1 Define a server named SYBASE. Its server class is *sql_server*, and its name in the *interfaces* file is SYBASE:

```
exec sp_addserver SYBASE, sql_server, SYBASE
```

2 Define a remote login alias. This step is optional. User "sa" is known to remote server SYBASE as user "sa," password "timothy":

```
exec sp_addexternlogin SYBASE, sa, sa, timothy
```

3 Define the remote authors table:

```
create existing table authors
 (
  au_id       id            not null,
  au_lname    varchar(40)   not null,
  au_fname    varchar(20)   not null,
  phone       char(12)      not null,
  address     varchar(40)   null,
  city        varchar(20)   null,
  state       char(2)       null,
  country     varchar(12)   null,
  postalcode  char(10)      null
 )
at "SYBASE.pubs2.dbo.authors"
```

4 Update statistics in tables to ensure reasonable choices by the query optimizer:

```
update statistics authors
```

5 Execute a query to test the configuration:

```
select * from authors where au_lname = 'Carson'
```

11.9.2 Remote procedures as proxy tables

In Chapter 2, "Understanding Component Integration Services," in the section "Topics and Issues," replace the section "Remote Procedures as Proxy Tables" with the following section.

Remote procedures as proxy tables

Component Integration Services users can map remote or external objects of the type procedure to local proxy tables. If a table is created in this way, it can be referenced only by the select and drop commands. The commands insert, delete, and update generate error messages, since the table is assumed to be read-only. Proxy definitions should only be created for procedures which return data.

If an object of the type procedure has been defined within the server, a query is not issued to the remote server on which the object resides. Instead, Component Integration Services issues an RPC and treats the results from the RPC as a read-only table.

An optional clause may be added to the create existing table statement to indicate the remote object is actually a stored (or other) procedure instead of a table. Without this clause, the remote object is assumed to be a table or view:

```
create existing table t1
    (
        column_1    int,
        column_2    int
    )
    EXTERNAL PROCEDURE AT "SERVER_A.mydb.dbo.myproc"
    go
    select * from myproc
```

When this query is issued, the server sends the RPC named *myproc* to server SERVER. Row results are treated like the results from any other table; they can be sorted, joined with other tables, grouped, inserted into another table, and so forth.

Component Integration Services attempts to pass as many of the search arguments as possible to the remote server, but depending on the SQL statement being executed, Component Integration Services might perform the result set calculation itself. Each parameter represents a search for an exact match, for example, the = operator.

The following are rules which define the parameters sent to the RPC. If an RPC will be used as a Component Integration Services object, these rules should be kept in mind during development.

- Component Integration Services sends = operators in the where clause as parameters. For example, the query:

  ```
  select * from rpc1 where a = 3 and b = 2
  ```

  results in Component Integration Services sending two parameters. Parameter a has a value of 3 and parameter b has a value of 2. The RPC is expected to return only result rows in which column a has a value of 3 and column b has a value of 2.

- Component Integration Services will not send any parameters for a where clause, or portion of a where clause, if there is not an exact search condition. For example:

  ```
  select * from rpc1 where a = 3 or b = 2
  ```

  Component Integration Services will not send parameters for a or b because of the or clause.

  Another example:

  ```
  select * from rpc1 where a = 2 and b < 3
  ```

  Component Integration Services will not send parameters because there is nothing in the where clause representing an exact search condition. Component Integration Services will perform the result set calculation locally.

In the case where the remote object is type *procedure*, several processing differences will occur:

- No indexes will be created for objects of this type.
- A column list must be provided which matches the description of the remote procedure's result set. This column list is the responsibility of the user, and no verification of its accuracy is provided.

Allowing the definition of remote procedures as local tables gives CIS the ability to treat the result set of a remote procedure as a 'virtual table,' which can be sorted, joined with other tables, or inserted into another table via insert/select syntax. However, tables of this type are considered read only:

- You cannot issue a delete, update, or insert command against a table of type *procedure.*
- You cannot issue a create index, truncate table, or alter table command against tables of this type.

Parameter columns

Column names beginning with underscore ('_') can be used to specify parameters, which are not part of the remote procedure's result set. These columns are referred to as parameter columns. For example:

```
create existing table rpc1
    (
        a    int,
        b    int,
        c    int,
        _p1  int null,
        _p2  int null
    )
    external procedure
    at "SYBASE.sybsystemprocs.dbo.myproc"

    select a, b, c from t1
    where _p1 = 10 and _p2 = 20
```

In this example, the parameter columns *_p1* and *_p2* are not expected in the result set, but can be referenced in the query. CIS will pass the search arguments to the remote procedure via parameters, using the names *@p1* and *@p2*.

If a parameter column is included in the select list, its value will be equivalent to the values specified for it in the where clause, if it was passed to the remote procedure as a parameter. If the parameter column did not appear in the where clause, or was not able to be passed to the remote procedure as a parameter, but was included in the select list, its value would be NULL.

A parameter column can be passed to the remote procedure as a parameter if it is what the Adaptive Server query processor considers to be a searchable argument. It is generally a searchable argument if it is not included in any or predicates. For example, the following query would prevent the parameter columns from being used as parameters:

```
select a, b, c from t1
where _p1 = 10 OR _p2 = 20
```

Rules exist for the definition of parameter columns in the create existing table statement:

- parameter columns must allow NULL.
- parameter columns cannot precede normal, result columns (i.e. they must appear at the end of the column list).

Allowing the definition of remote procedures as local tables gives CIS the ability to treat the result set of a remote procedure as a 'virtual table,' which can be sorted, joined with other tables, or inserted into another table via insert/select syntax. However, tables of this type are considered read only:

- You cannot issue a delete, update, or insert command against a table of type *procedure*;
- You cannot issue a create index, truncate table, or alter table command against tables of this type.

11.9.3 Sending text as RPC parameters

In Chapter 3, "Using Component Integration Services," in the section "RPC Handling and Component Integration Services" replace the section "Sending Text as RPC Parameters" with the following section.

Sending text as RPC parameters

You can send large blocks of text data as parameters to remote procedures. This is possible when the RPC is being handled by Component Integration Services, and a text pointer is provided as a parameter to the RPC. To distinguish the text pointer from an ordinary binary(16) parameter type, the thread property *textptr_paramters* must be set:

```
set textptr_parameters ON
```

In addition, the origin of the text pointer must be identified by a char or varchar parameter that contains the name of the table and column from which the *textptr* was derived. For example:

```
declare @pathname varchar(90)
declare @textptr1 binary(16)
declare @textptr2 binary(16)

select @pathname = "mydatabase.dbo.t1",
       @textptr1 = textptr(c1),
       @textptr2 = textptr(c2)
```

```
        from mydatabase.dbo.t1
        where ...

set textptr_parameters ON

set cis_rpc_handling ON

exec NETGW...myrpc @pathname, @textptr1, @textptr2

set textptr_parameters OFF
```

When the RPC named *myrpc* gets sent to server NETGW, the *@pathname* parameter is not actually sent, but is used to help locate table from which the text pointers were derived.

The *@pathname* parameter must immediately precede the binary(16) text pointers, otherwise it will be considered an ordinary parameter and will be transmitted as such. If the *@pathname* parameter is not provided, the text pointers are treated as ordinary binary(16) parameters.

When the RPC is being prepared for transmission to the remote server, the text pointers are expanded into text chunks of no more than 32K bytes in size:

- Each of these text chunks is passed as a separate parameter whose Open Client datatype is CS_LONGCHAR_TYPE.
- Text chunks are created as required to contain the entire text represented by the text pointer.
- More than one text pointer can be handled in this manner within a single RPC.
- If all text pointers are derived from the same table, then all can be preceded by a single *@pathname* parameter.
- If text pointers are derived from different tables, a separate *@pathname* parameter must introduce each text pointer or group of text pointers. This indicates the table from which the text pointers have been obtained.
- The current value of *@@textsize* will be ignored; all the text from each text pointer will be transmitted in this manner.

This scheme also works with proxy tables mapped to remote procedures. For example:

```
create existing table myrpctable
    (
        id       int,           -- result column
        crdate   datetime,      -- result column
```

```
    name  varchar(30),      -- result column
    _pathname  varchar(90), -- parameter column
    _textptr1  binary(16),  -- parameter column
    _textptr2  binary(16),  -- parameter column

    external procedure at 'NETGW...myrpc'
    go

declare  @textptr1  binary(16)
declare  @textptr2  binary(16)

select   @textptr1 = textptr(c1),
         @textptr2 = textptr(c2)
from  mydatabase.dbo.t1
where  <whatever>

set textptr_parameters ON
select  id,
    crdate,
    name
from  myrpctable
where  _pathname = "mydatabase.dbo.t1" and
    _textptr1 = @textptr1 and
    _textptr2 = @textptr2
```

This capability is only activated if the server to which the RPC is being sent supports the Open Client datatype CS_LONGCHAR_TYPE. Support for this datatype is determined by inquiring datatype capabilities using the Open Client function ct_capabiliites().

11.10 Online Help for ASE plug-in to Sybase Central

The Help topic "Creating a Primary Key" is not correct. The correct text is:

To create a primary key:

1 Open the table editor.

2 Select a column on which you want to create a primary key.

3 Choose Edit Primary-key Columns from the popup menu.

4 Complete the Edit Primary-key Columns dialog box.

You cannot create a primary key on a column that allows nulls.

The primary key is created in the Unique Constraints folder.

12. 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.

13. Other sources of information

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

- Technical Library CD contains product manuals and technical documents and is included with your software. The DynaText browser (included on the Technical Library CD) allows you to access technical information about your product in an easy-to-use format.

  Refer to the *Technical Library Installation Guide* in your documentation package for instructions on installing and starting the Technical Library.

- Technical Library Product Manuals Web site is an HTML version of the Technical Library CD that you can access using a standard Web browser. In addition to product manuals, 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 Product Manuals Web site, go to Product Manuals at http://www.sybase.com/support/manuals/.

13.1 Sybase certifications on the Web

Technical documentation at the Sybase Web site is updated frequently.

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

1 Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.

2 Select a product from the product pick list and click Go.

3 Select the Certification Report filter, specify a time frame, and click Go.

4 Click a Certification Report title to display the report.

❖ **For the latest information on EBFs and Updates**

1 Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.

2 Select EBFs/Updates. Enter user name and password information, if prompted (for existing web accounts) or create a new account (a free service).

3 Specify a time frame and click Go.

4 Select a product.

5 Click an EBF/Update title to display the report.

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

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

1 Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/

2 Click MySybase and create a MySybase profile.