

CHAPTER 4

Database Administration Utilities

About this chapter Adaptive Server Anywhere includes a set of utility programs for backing up databases and performing other database administration tasks. This chapter provides reference information for each of the database administration utilities.

Contents

Topic	Page
Administration utilities overview	65
The Backup utility	67
The Collation utility	71
The Compression utility	75
The Console utility	77
The DBPing utility	78
The DBSpawn utility	79
The Erase utility	80
The Information utility	82
The Initialization utility	84
The Interactive SQL utility	91
The Log Transfer Manager	93
The Log Translation utility	98
The REBUILD component	102
The Stop utility	103
The Transaction Log utility	105
The Uncompression utility	108
The Unload utility	110
The Upgrade utility	116
The Validation utility	119
The Write File utility	121

Administration utilities overview

This chapter presents reference information on the programs and database administration utilities that are part of Adaptive Server Anywhere. The utilities can be accessed from Sybase Central, from Interactive SQL, or as command-line programs.

☞ For comprehensive documentation on Sybase Central, see the Sybase Central online Help. For an introduction to the Sybase Central database administration tool, see "Managing Databases with Sybase Central" on page 45 of the book *First Guide to SQL Anywhere Studio*.

The administration utilities use a set of system environment variables. These variables are described in "Environment variables" on page 5.

The administration utilities also use a standard set of return codes. These return codes are described in "Software component Return codes" on page 65.

Adaptive Server Anywhere for Unix platforms

The syntax for starting utilities on Unix platforms differs slightly from other platforms. On Unix platforms, programs are executed with commands typed in the lower case, and program files do not have an *.exe* suffix.

File administration statements

A set of SQL statements are available that carry out some of the tasks that the administration utilities carry out. These statements are listed in "SQL Statements" on page 339.

Software component Return codes

All database components use the following return codes. The header file *sqldef.h* has constants defined for these codes.

Code	Explanation
0	Success
1	General failure
2	Invalid file format, and so on
3	File not found, unable to open, and so on
4	Out of memory
5	Terminated by the user
6	Failed communications

Code	Explanation
7	Missing a required database name
8	Client/server protocol mismatch
9	Unable to connect to the database server
10	Database server not running
11	Database server not found
254	Reached stop time
255	Invalid parameters on the command line

The Backup utility


With the Backup utility, you can back up running databases, database files, transaction logs, and write files.

You can access the Backup utility in the following ways:

- ◆ From Sybase Central, for interactive use under Windows 95 or Windows NT.
- ◆ From the system command line, using the *dbbackup* utility. This is useful for incorporating backup procedures into batch or command files.

The Backup utility makes a backup copy of all the files for a single database. A simple database consists of two files: the main database file and the transaction log. More complicated databases can store tables in multiple files, with each file as a separate dbspace. All backup filenames are the same as the database filenames.

Running the Backup utility on a running database is equivalent to copying the database files when the database is not running. Thus, you can back up the database while other applications or users are using it.

 For a description of suggested backup procedures, see "Backup and Data Recovery" on page 553 of the book *Adaptive Server Anywhere User's Guide*.


Backing up a database from Sybase Central

❖ **To back up a running database:**

- 1 Connect to the database.
- 2 Right-click the database and click Backup in the popup menu. The Backup wizard is displayed.
- 3 Follow the instructions in the wizard.

❖ **To back up a database file or a running database:**

- 1 Open the Database Utilities folder in the left panel.
- 2 Double-click Backup Database in the right panel.
- 3 Follow the instructions in the wizard.

 For full information on backing up a database from Sybase Central, see the Sybase Central online Help. For more information about options, see "Backup utility options" on page 68

The DBBACKUP command-line utility

Syntax `dbbackup [switches] directory`

Windows 3.x syntax `dbbackw [switches] directory`

Switch	Description
<code>-c "keyword=value; ..."</code>	Supply database connection parameters
<code>-d</code>	Only back up the main database file
<code>-l file</code>	Live backup of the transaction log to a file
<code>-n</code>	Change the naming convention for the back up transaction log
<code>-o file</code>	Log output messages to a file
<code>-q</code>	Quiet mode—do not print messages
<code>-r</code>	Rename and start a new transaction log
<code>-t</code>	Only back up the transaction log
<code>-w</code>	Only back up the write file
<code>-x</code>	Delete and restart the transaction log
<code>-y</code>	Replace files without confirmation

Description If none of the switches `-d`, `-t`, or `-w` are used, all database files are backed up.

☞ For more information about the command-line switches, see "Backup utility options" on page 68.

Backup utility options

Directory The directory to which the backup files are copied. If the directory does not exist, it is created. However, the parent directory must exist.

Connection parameters (-c) For a description of the connection parameters, see "Connection parameters" on page 46 of the book *Adaptive Server Anywhere User's Guide*. If the connection parameters are not specified, connection parameters from the SQLCONNECT environment variable are used, if set. The **user ID** must have DBA authority.

For example, the following statement backs up the **asademo** database running on the server **sample_server**, connecting as user ID **DBA** with password **SQL**:

```
dbbackup -c  
"eng=sample_server;dbn=asademo;uid=dba;pwd=sql"
```

Backup main database only (-d) Back up the main database files only, without backing up the transaction log file or a write file, if one exists.

Live backup (-l lower-case L) This option is provided to enable a secondary system to be brought up rapidly in the event of a server crash. A live backup does not terminate, but continues running while the server runs. It runs until the primary server crashes. At that point, it is shut down, but the backed up log file is intact and can be used to bring a secondary system up quickly.

Change backup transaction log naming convention (-n) This option changes the naming convention of the backup transaction log file to *yymmddxx.log*, where *xx* is a number from 00 to 99 and *yymmdd* represents the current year, month and day. By default, the name used for the backup transaction log file is identical to the file name of the transaction log that is being backed up.

When using the *-n* option, you must also use the *-r* or *-t* options.

The two-digit year notation does not cause any year 2000 problems. The names are used solely for identification, not for ordering.

Output messages to log (-o) Sends backup messages to a named log file.

Operate quietly (-q) Do not display messages on a window. This option is available only from the command-line utility.

Rename and start new transaction log (-r) This option forces a checkpoint and the following three steps to occur:

- ◆ **Step 1** The current working transaction log file is copied and saved to the directory specified in the command line.
- ◆ **Step 2** The current transaction log remains in its current directory, but is renamed using the format *yymmddxx.log*, where *xx* is a number from 00 to 99 and *yymmdd* represents the current year, month and day. This file is then no longer the current transaction log.
- ◆ **Step 3** A new transaction log file is generated that contains no transactions. It is given the name of the file that was previously considered the current transaction log, and is used by the database server as the current transaction log.

Back up the transaction log file only (-t) This can be used as an incremental backup since the transaction log can be applied to the most recently backed up copy of the database file(s).

Back up the database write file only (-w) For a description of database write files, see "The Write File utility" on page 121.

Delete and restart the transaction log (-x) With this option, the existing transaction log is backed up, then the original is deleted and a new transaction log is started with the same name. This option causes the backup to wait for a point when all transactions from all connections are committed.

Operate without confirming actions (-y) Without this option, you are prompted to confirm the creation of the backup directory or the replacement of a previous backup file in the directory.

The Collation utility

With the Collation utility, you can extract a collation (sorting sequence) from the SYS.SYSCOLLATION system table of a database into a file that is suitable for creating a database using a custom collation.


The file that is produced can be modified and used with Sybase Central or the `-z` option of `dbinit` to create a new database with a custom collation.

Be sure to change the label on the line that looks like:

```
Collation label (name)
```

Otherwise, the collation cannot be used to create a database.

If you wish to create a custom collation but have not yet created a database, extract a collation from the sample database.

 For more information on custom collating sequences, see "Creating databases with custom collations" on page 309 of the book *Adaptive Server Anywhere User's Guide*.


You can access the Collation utility in the following ways:

- ◆ From Sybase Central.
- ◆ From the system command line, using the `dbcollat` command-line utility.

Extracting a collation using Sybase Central

❖ To use the Collation utility from Sybase Central:

- 1 Open the Database Utilities folder in the left panel.
- 2 Double-click Create a Custom Collation in the right panel. The Custom Collation Wizard is displayed.
- 3 Follow the instructions in the Wizard.

 For full information on compressing a database in Sybase Central, see the Sybase Central online Help.

The DBCOLLAT command-line utility


Syntax `dbcollat [switches] output-file`

Windows 3.x syntax `dbcollw [switches] output-file`

Switch	Description
-c "keyword=value; ..."	Supply database connection parameters
-d filename	Convert the definition file to INSERT statements
-e	Include empty mappings
-o filename	Output log messages to a file
-q	Quiet mode — do not print messages
-x	Use hex for extended characters (7F-FF)
-y	Replace the file without confirmation
-z col-seq	Specify a collating sequence label

Description

The DBCOLLAT command line utility allows you to extract a collation (sorting sequence) from the SYS.SYSCOLLATION system table of a database into a file that is suitable for creating a database using a custom collation.

 For more information about the command-line switches, see "Collation utility options" on page 72.

Collation utility options

Connection parameters (-c) For a description of the connection parameters, see "Connection parameters" on page 46 of the book *Adaptive Server Anywhere User's Guide*. If the connection parameters are not specified, connection parameters from the SQLCONNECT environment variable are used, if set.

For example, the following statement extracts a collation file from the **asademo** database that is running on the **sample_server** server, and connects as user ID **DBA** with password **SQL**:

```
dbcollat -c
"eng=sample_server;dbn=asademo;uid=dba;pwd=sql"
c:\sample\col
```

Convert the definition file to an INSERT statement (-d) A database is created using the CREATE DATABASE statement. The Initialization utility generates this statement internally. You can specify which collation to use in the CREATE DATABASE statement.

The collation definitions exist as an INSERT statement in a file named *collseqs.sql* in the *scripts* subdirectory of your installation directory. You can add additional custom collations to *custom.sql*, also in the *scripts* subdirectory.

This option provides a way to extract a collation definition from an existing database, as a starting point to creating a custom collation. It converts a custom collation definition file into an INSERT statement for SYSCOLLATION, suitable for adding into *custom.sql*.

From the *dbcollat* command-line utility you can use this option as follows:

```
dbcollat -d coll-defn-file output-file
```

The file *coll-defn-file* is read and parsed as a collation definition, and an INSERT statement is output to *output-file*.

Include empty mappings (-e) Normally, collations don't specify the actual value that a character is to sort to. Instead, each line of the collation sorts to one position higher than the previous line. However, older collations have gaps between some sort positions. Normally, the Collation utility skips the gaps and writes the next line with an explicit sort-position. This option causes the Collation utility to write empty mappings (consisting of just a colon (:)) for each line in the gap.

Output log messages to a file (-o) Redirect the log messages from the Collation utility to a named file.

Operate quietly (-q) Do not display messages on a window. This option is available only from the command-line utility.

Use hexadecimal for extended characters [7F to FF] (-x) Extended single-byte characters (whose value is greater than hex 7F) may or may not appear correctly on your screen, depending on whether the code page on your computer is the same as the code page being in the collation that you are extracting. This option causes the Collation utility to write any character to hex 7F or above as a two-digit hexadecimal number, in the form `\xdd`. For example:

```
\x80, \xFE
```

Without the `-x` option, only characters from hex 00 to hex 1F, hex 7F and hex FF are written in hexadecimal form.

Operate without confirming actions (-y) Without this option, you are prompted to confirm replacing an existing collation file.

Specify a collating sequence label (-z) Specify the label of the collation to be extracted. The names of the collation sequences can be found in the **collation_label** column of the SYS.SYSCOLLATION table. If this option is not specified, the Collation utility extracts the collation label that is being used by the database.

The Compression utility

With the Compression utility you can compress a database file. The Compression utility reads the given database file and creates a compressed database file. Compressed databases are usually 40 to 60 per cent of their original size. The database server cannot update compressed database files: they must be used in conjunction with write files.

The Compression utility does not compress files other than the main database file.


You can access the Compression utility in the following ways:

- ◆ From Sybase Central, for interactive use under Windows 95 or NT.
- ◆ From the system command line, using the *dbshrink* command-line utility. This is useful for incorporating into batch or command files.

Compressing a database in Sybase Central

❖ **To compress a database file:**

- 1 Open the Database Utilities folder in the left panel.
- 2 Double-click Compress Database in the right panel. The Compress Database wizard is displayed.
- 3 Follow the instructions in the wizard.

 For full information on compressing a database in Sybase Central, see the Sybase Central online Help.

The DBSHRINK command-line utility

Syntax

dbshrink [*switches*] *database-file* [*compressed-database-file*]

**Windows 3.x
syntax**

dbshrinw [*switches*] *database-file* [*compressed-database-file*]

Switch	Description
-q	Quiet mode—do not print messages
-y	Erase an existing output file without confirmation

Description

dbshrink reads the given *database* file and creates a compressed database file. The compressed filename defaults to the same name with an extension of *cdb*. The output filename (with extension) must not be same as the input filename (with extension).

☞ For more information about the command-line switches, see "Compression utility options" on page 76.

Compression utility options

Operate quietly (-q) Do not display messages on a window. This option is available only from the command-line utility.

Operate without confirming actions (-y) Without this option, you are prompted to confirm the replacement of an existing database file.

The Console utility

Syntax `dbconsol [switches]`


Windows 3.x syntax `dbconsolw [switches]`

Switch	Description
<code>-c "keyword=value; ..."</code>	Supply database connection parameters

Description Start the server console utility.

The server console utility allows you to monitor the server from a client machine. You can use it to track who is logged on to a database server elsewhere on your network. You can also display both the server and the client statistics on your local client screen, disconnect users and configure the database server.

Switches `-c "keyword=value; ..."` Specify connection parameters.

 For a description of connection parameters, see "Connection parameters" on page 46 of the book *Adaptive Server Anywhere User's Guide*.

The DBPing utility

Syntax `dbping [switches]`

Windows 3.x syntax `dbpingw [switches]`

Switch	Description
<code>-c "keyword=value; ..."</code>	Database connection parameters
<code>-o</code>	Output log messages to a file
<code>-q</code>	Operate quietly—do not print messages
<code>-d</code>	Make a database connection if the server is found

Description The *dbping* command-line utility is provided to help debug connection problems. It takes a full or partial connection string and returns a message indicating whether the attempt to locate a server or database, or to connect, was successful.

Switches

- `-o` Log output messages to a file.
- `-q` Operate in quiet mode. If *dbping* fails, a message is always displayed. If *dbping* succeeds, no message is displayed if `-q` is specified.

The `-d` switch is used for specifying that you want to ping the database, not just the server. If the `-d` switch is not supplied, then *dbping* is successful if it finds the server specified by the `-c` switch. If the `-d` switch is supplied, then *dbping* is successful only if it can connect to the server and also connect to a database.

For example, if I have an engine named "verna" running on the database "sample" the following will succeed:

```
dbping -c "eng=verna;dbn=somedb"
```

If I had included the `-d` switch, however, it would have failed with the message "Ping database failed -- specified database not found":

Essentially what we do to find the engine is exactly what we do in a string connect. (So, for example, the dbcli6 Connection Parameters are considered in the same way.) If the `-d` switch is not specified, then we stop as soon as we find the server, and return success to the user. If the `-d` switch is specified, then we do the same thing as `db_string_connect`. In this case, we return success to the user if we did not get an error connecting (warnings are tolerated).

The DBSpawn utility

Syntax `dbspawn [-v] [server command line]`

Switch	Description
-v	Display error message in the event of failure

Description The DBSPAWN command-line utility is provided to start a server in the background. DBSPAWN will start the server in the background and will return with an exit code of 1 (success) or 0 (failure). If the server specified in "server command line" is already running, dbspawn will report failure. The -v option will cause the error message (in case of failure) to be displayed.

Switches The -v switch specifies that you want to receive an error message if the DBSPAWN command fails.

The Erase utility

With the Erase utility, you can erase a database file or write file and its associated transaction log, or you can erase a transaction log file or transaction log mirror file. All database files, write files, and transaction log files are marked read-only to prevent accidental damage to the database and accidental deletion of the database files. Deleting a database file that references other dbspaces does not automatically delete the dbspace files.

If you erase a database file or write file, the associated transaction log and transaction log mirror are also deleted. If you erase a transaction log for a database that also maintains a transaction log mirror, the mirror is not deleted.

You can access the Erase utility in the following ways:

- ◆ From Sybase Central, for interactive use under Windows 95 or NT.
- ◆ From the system command line, using the *dberase* command-line utility. This is useful for incorporating into batch or command files.

Erasing a database from Sybase Central

❖ **To erase a database file:**

- 1 Open the Database Utilities folder in the left panel.
- 2 Double-click Erase Database in the right panel. The Erase a Database wizard is displayed.
- 3 Follow the instructions in the wizard.

🔗 For full information on erasing a database from Sybase Central, see the Sybase Central online Help.

The DBERASE command-line utility

Syntax

dberase [*switches*] *database-file*

**Windows 3.x
syntax**

dberasew [*switches*] *database-file*

Switch	Description
-o	Output log messages to a file
-q	Operate quietly—do not print messages
-y	Erase files without confirmation

Description

The *database-file* may be a database file, write file, or transaction log file. The full filename must be specified, including extension. If a database file or write file is specified, the associated transaction log file (and mirror, if one is maintained) is also erased.

↪ For more information about the command-line switches, see "Erase utility options" on page 81.

Erase utility options

Output log messages to file (-o) Redirect log messages to the named file. Do not display messages on a window. This option is available only from the command-line utility.

Operate without confirming actions (-y) Without this option, you are prompted to confirm the deletion of each file.

The Information utility

With the Information utility, you can display information about a database file or write file. The database should not be running when you run the Information utility. The utility indicates when the database was created, the name of any transaction log file or log mirror that is maintained, the page size, and other information. Optionally, it can also provide table usage statistics and details.

You can access the Information utility in the following ways:

- ◆ From the system command line, using the *dbinfo* command-line utility.

The DBINFO command-line utility

Syntax `dbinfo [switches] filename`
Windows 3.x syntax `dbinfo [switches] filename`

Switch	Description
-c " <i>keyword=value; ...</i> "	Database connection parameters
-o <i>filename</i>	Output log messages to a file
-q	Suppress any messages
-u	Output page usage statistics

Description The DBINFO command-line utility allows you to display information about a database file or write file. It indicates when the database was created, the name of any transaction log file or log mirror that is maintained, the page size, and other information. Optionally, it can also provide table usage statistics and details.

☞ For more information about the command-line switches, see "Information utility options" on page 82.

Information utility options

Connection parameters (-c) Specify connection parameters. See "Connection parameters" on page 46 of the book *Adaptive Server Anywhere User's Guide* for a description of the connection parameters.

Any valid user ID can run the Information utility, but to obtain page usage statistics you need DBA authority.

Output log messages to a file (-o) Redirect log messages to the named file.

Operate quietly (-q) Do not display the information. The return code may still provide useful information (see "Software component Return codes" on page 65). This option is available only from the command-line utility.

Page usage statistics (-u) Display information about the usage and size of all tables, including system and user-defined tables.

You can only request page usage statistics if no other users are connected to the database.

The Initialization utility

With the Initialization utility, you can initialize (create) a database. A number of database attributes are specified at initialization and cannot be changed later except by unloading, reinitializing, and rebuilding the entire database:

- ◆ Case sensitivity or insensitivity
- ◆ Storage as an encrypted file
- ◆ Treatment of trailing blanks in comparisons
- ◆ The page size
- ◆ The collation sequence

In addition, the choice of whether to use a transaction log and a transaction log mirror is made at initialization. This choice can be changed later using the transaction log utility.

You can access the Initialization utility in the following ways:

- ◆ From Sybase Central, for interactive use under Windows 95 or NT.
- ◆ From the system command line, using the *dbinit* command-line utility. This is useful for incorporating into batch or command files.

The personal server is required by the initialization utility and must be running during the initialization process.

Creating a database in Sybase Central

❖ To create a database:

- 1 Open the Database Utilities folder in the left panel.
- 2 Double-click Create Database in the right panel. The Database Creation wizard is displayed.
- 3 Follow the instructions in the wizard.

🔗 For full information on creating a database in Sybase Central, see the Sybase Central online Help.

The DBINIT command-line utility

Syntax `dbinit [switches] new-database-file`

**Windows 3.x
syntax****dbinitw** [*switches*] *new-database-file*

Switch	Description
-b	Blank padding of strings for comparisons
-c	Case sensitivity for all string comparisons
-e	Encrypt the database
-i	Do not install Sybase jConnect support
-j	Do not install Sybase runtime Java classes
-k	Omit Watcom SQL compatibility views SYS.SYSCOLUMNS and SYS.SYSINDEXES
-l	List the available collating sequences
-m <i>file-name</i>	Use a transaction log mirror (default is no mirror)
-n	No transaction log
-o <i>filename</i>	Output messages to file
-p <i>page-size</i>	Set page size
-q	Quiet mode—do not print messages
-t <i>log-name</i>	Transaction log filename (default is the database name with <i>.log</i> extension)
-z <i>col-seq</i>	Collation sequence used for comparisons

Description

For example, the database *test.db* can be created with 1024 byte pages as follows:

```
dbinit -p 1024 test.db
```

For more information about the command-line switches, see "Initialization utility options" on page 85.

Initialization utility options

Ignore trailing blanks (-b) Trailing blanks are ignored for comparison purposes, and embedded SQL programs pad strings that are fetched into character arrays. For example, the two strings

```
'Smith'
'Smith   '
```

would be treated as equal in a database that ignores trailing blanks.

This option is provided for compatibility with the ISO/ANSI SQL standard, which is to ignore trailing blanks in comparisons. The default is that blanks are significant for comparisons.

Case sensitivity for all string comparisons (-c) For databases created with this option, all values are considered to be case sensitive in comparisons and string operations.

This option is provided for compatibility with the ISO/ANSI SQL standard. The default is that all comparisons are case insensitive.

User ID and password

All databases are created with at least one user ID, **DBA**, with password **SQL**. If you create a database that requires case-sensitive comparisons, the **DBA** user ID and its password must be entered in upper case.

Encrypt the database (-e) Encryption makes it more difficult for someone to decipher the data in your database by using a disk utility to look at the file. File compaction utilities cannot compress encrypted database files as much as unencrypted ones.

Do not install Sybase jConnect support (-i) If you wish to use the Sybase jConnect JDBC driver to access system catalog information, you need to install jConnect support. Use this option if you wish to exclude the jConnect system objects. You can still use JDBC, as long as you do not access system information.

Do not install Sybase runtime Java classes (-j) If you wish to use Java in your database, you must install the Sybase runtime Java classes. By default, these classes are installed. The runtime classes add several megabytes to the size of a database, so if you do not intend to use Java classes, you can specify the `-j` switch to avoid installing them.

Omit Watcom SQL compatibility views (-k) By default, the *dbinit* command-line utility generates the views SYS.SYSCOLUMNS and SYS.SYSINDEXES for compatibility with system tables that were available in Watcom SQL (versions 4 and earlier of this software). These views will conflict with the Sybase Adaptive Server Enterprise compatibility views DBO.SYSCOLUMNS and DBO.SYSINDEXES unless the `-c` case sensitivity command-line switch is used.

List the available collating sequences (-l) *dbinit* lists the available collating sequences and then stops. No database is created. The list of available collating sequences is automatically presented in Sybase Central and in the Interactive SQL Database Tools window.

Use a transaction log mirror (-m) A transaction log mirror is an identical copy of a transaction log, usually maintained on a separate device, for greater protection of your data. By default, Adaptive Server Anywhere does not use a mirrored transaction log.

Do not use a transaction log (-n) Creating a database without a transaction log saves disk space. The transaction log is required for data replication and provides extra security for database information in case of media failure or system failure.

Output log messages to file (-o) Redirect log messages to the named file.

Page size (-p) The page size for a database can be 512, 1024, 2048 or 4096 bytes, with 1024 being the default. Any other value for the size will be changed to the next larger size. Large databases usually benefit from a larger page size.

Operate quietly (-q) Do not display messages on a window. This option is available only from the command-line utility.

Set the transaction log filename (-t) The transaction log is a file where the database server logs all changes, made by all users, no matter what application system is being used. The transaction log plays a key role in backup and recovery (see "The transaction log" on page 557 of the book *Adaptive Server Anywhere User's Guide*), and in data replication. If the filename has no path, it is placed in the same directory as the database file. If you run *dbinit* without specifying `-t` or `-n`, a transaction log is created with the same filename as the database file, but with extension *log*.

Collating sequence or collation filename (-z) The collation sequence is used for all string comparisons in the database.

The database is created with all of the collations that are listed in SYS.SYSCOLLATION.

If you want to use a custom collation, use the Collation utility to extract the closest collation from the database. You can then modify the collation and add it to the file *custom.sql* in the *scripts* subdirectory of your installation directory. Finally, you use the Initialization utility to create the database and specify the new collation.

You must change the collation label in the custom collation file. Otherwise, the Initialization utility prevents the insertion of the new collation, since it will be a duplicate label.

🔗 For more information on custom collating sequences, see "Database Collations and International Languages" on page 289 of the book *Adaptive Server Anywhere User's Guide*.

In order to change the collation that an existing database uses, it is necessary to unload the database, create a new database using the appropriate collation, then reload the database.

If `-z` is specified and names a collation, that collation will be used by the database. If `-z` is specified but does not name a collation, it is assumed that the name is a filename, and the file will be opened and the collation will be parsed from the file. The specified collation will then be used by the database.

If `-z` is not specified, the default collation is used. Normal ASCII (binary) ordering is used for the lower 128 characters. For the upper 128 characters (also called the extended characters), any character that is an accented form of a letter in the lower 128 are sorted to the same position as the unaccented form. The determination of whether or not an extended character is an accented letter is based upon code page 850 (multilingual code page).

The following table identifies the available collating sequence labels. All collating sequences except EBCDIC do not affect the normal ASCII character set (lower 128).

Collation name	Type	Description
internal	OEM	Based on Code page 850
437	OEM	Code page 437, ASCII, United States
850	OEM	Code page 850, ASCII, Multilingual
852	OEM	Code page 852, ASCII, Slavic/Latin II
860	OEM	Code page 860, ASCII, Portugal
863	OEM	Code page 863, ASCII, Canada-French
865	OEM	Code page 865, ASCII, Norway
EBCDIC	ANSI	EBCDIC
437EBCDIC	OEM	Code Page 437, EBCDIC
437LATIN1	OEM	Code Page 437, Latin 1
437ESP	OEM	Code Page 437, Spanish
437SVE	OEM	Code Page 437, Swedish/Finnish
819CYR	ANSI	Code Page 819, Cyrillic
819DAN	ANSI	Code Page 819, Danish
819ELL	ANSI	Code Page 819, Greek
819ESP	ANSI	Code Page 819, Spanish
819ISL	ANSI	Code Page 819, Icelandic
819LATIN1	ANSI	Code Page 819, Latin 1

Collation name	Type	Description
819LATIN2	ANSI	Code Page 819, Latin 2
819NOR	ANSI	Code Page 819, Norwegian
819RUS	ANSI	Code Page 819, Russian
819SVE	ANSI	Code Page 819, Swedish/Finnish
819TRK	ANSI	Code Page 819, Turkish
850CYR	OEM	Code Page 850, Cyrillic
850DAN	OEM	Code Page 850, Danish
850ELL	OEM	Code Page 850, Greek
850ESP	OEM	Code Page 850, Spanish
850ISL	OEM	Code Page 850, Icelandic
850LATIN1	OEM	Code Page 850, Latin 1
850LATIN2	OEM	Code Page 850, Latin 2
850NOR	OEM	Code Page 850, Norwegian
850RUS	OEM	Code Page 850, Russian
850SVE	OEM	Code Page 850, Swedish/Finnish
850TRK	OEM	Code Page 850, Turkish
852LATIN2	OEM	Code Page 852, Latin 2
852CYR	OEM	Code Page 852, Cyrillic
855CYR	OEM	Code Page 855, Cyrillic
856HEB	OEM	Code Page 856, Hebrew
857TRK	OEM	Code Page 857, Turkish
860LATIN1	OEM	Code Page 860, Latin 1
861ISL	OEM	Code Page 861, Icelandic
862HEB	OEM	Code Page 862, Hebrew
863LATIN1	OEM	Code Page 863, Latin 1
865NOR	OEM	Code Page 865, Norwegian
866RUS	OEM	Code Page 866, Russian
869ELL	OEM	Code Page 869, Greek
920TRK	ANSI	Code page 920, Turkish
SJIS	ANSI	Japanese Shift-JIS Encoding
SJIS2	ANSI	Japanese Shift-JIS Encoding, Sybase Adaptive Server Enterprise-compatible

Collation name	Type	Description
EUC_JAPAN	ANSI	Japanese EUC JIS X 0208-1990 and JIS X 0212-1990 Encoding
EUC_CHINA	ANSI	Chinese GB 2312-80 Encoding
EUC_TAIWAN	ANSI	Taiwanese Big 5 Encoding
EUC_KOREA	ANSI	Korean KS C 5601-1992 Encoding
UTF8	ANSI	UCS-4 Transformation Format
ISO_1	ANSI	ISO8859-1, Latin 1
ISO_BINENG	ANSI	Binary ordering, English ISO/ASCII 7-bit letter case mappings
WIN_LATIN1	ANSI	Windows Latin 1, similar to ISO_1, ISO8859-1, with extensions
WIN_LATIN5	ANSI	A superset of ISO 8859-9 with characters also in 80-9F

The Interactive SQL utility

Syntax `dbisql [switches] [dbisql-command | command-file]`

Windows 3.x syntax `dbisqlw [switches] [dbisql-command | command-file]`

Switch	Description
<code>-c "keyword=value; ..."</code>	Supply database connection parameters
<code>-d delimiter</code>	Specify command delimiter
<code>-q</code>	Quiet mode—no windows or messages
<code>-x</code>	Syntax check only—no commands executed

Description Interactive SQL provides the user with an interactive environment for database browsing and for sending SQL statements to the database server.

Interactive SQL allows you to type SQL commands, or run command files. It also provides feedback about the number of rows affected, the time required for each command, the execution plan of queries, and any error messages.

If *dbisql-command* is specified, Interactive SQL executes the command. You can also specify a command file name. If no *dbisql-command* is specified, SQL enters interactive mode, where you can type a command into a command window.

The Windows 3.x executable name is *dbisqlw*. You can set up Program Manager icons to start *dbisqlw* with the same command line syntax as specified above.

Switches `-c "keyword=value; ..."` Specify connection parameters. See "Connection parameters" on page 46 of the book *Adaptive Server Anywhere User's Guide* for a description of the connection parameters. If this option is not specified, the environment variable SQLCONNECT is used. If required connection parameters are not specified, you are presented with a dialog where you can enter the connection parameters.

`-d` Specify a command delimiter. By default, the delimiter is the semi-colon.

`-q` Operate quietly. Interactive SQL does not display any information on the screen. This is useful only if you start Interactive SQL with a command or command file.

`-x` Scan commands but do not execute them. This is useful for checking long command files for syntax errors.

☞ For or detailed descriptions of SQL statements and Interactive SQL commands, see "SQL Language Elements" on page 179.

Examples

- ◆ The following command, entered at a system prompt, runs the command file *mycom.sql* against the current default server, using the user ID **DBA** and the password **SQL**.

```
dbisql -c "uid=dba;pwd=sql" mycom.sql
```

- ◆ The following command, entered at a system prompt, adds a user to the current default database:

```
dbisql -c "uid=dba;pwd=sql" grant connect to joe  
identified by passwd
```

Starting Interactive SQL from Sybase Central

You can start Interactive SQL from Sybase Central in the following ways:

- ◆ Right-click a database, and select Open Interactive SQL from the popup menu.
- ◆ Right-click a table, and select View Data from the popup menu. Interactive SQL opens with the data in the table displayed in the Data window.
- ◆ Right-click a stored procedure, and select Test from the popup menu. Interactive SQL opens with a test script in the Command window.

The Log Transfer Manager

Syntax `dbltn [switches]`

Parameters	Switch	Description
	<code>-C config_file</code>	Use the named configuration file.
	<code>-I interface_file</code>	Use the named interface file.
	<code>-S LTM_name</code>	Specify an LTM name.
	<code>-A</code>	Do not filter updates.
	<code>-M</code>	Recovery mode.
	<code>-dl</code>	Display log messages on screen
	<code>-o file</code>	Log output messages to file
	<code>-ot file</code>	Truncate file, and log output messages to file
	<code>-q</code>	Run in minimized window
	<code>-s</code>	Show Log Transfer Language (LTL) commands.
	<code>-v</code>	Verbose mode

Description The Log Transfer Manager (LTM) is also known as a **replication agent**.

The LTM is required for any Adaptive Server Anywhere database that participates in a Replication Server installation as a primary site. The Adaptive Server Anywhere LTM reads a database transaction log and sends committed changes to Replication Server. The LTM is not required at replicate sites.

The LTM sends messages to Replication Server in a language named Log Transfer Language (LTL).

By default, the LTM uses a log file named *DBLTM.LOG* to hold status and other messages. You can use the command-line options to change the name of this file and to change the volume and type of messages that are sent to it.

Switches

-C config_file Use the configuration file *config_file* to determine the LTM settings. The default configuration file is *dbltn.cfg*. The format of the configuration file is described in "The LTM configuration file" on page 94.

-I interfaces_file (Upper case i.) Use the named interfaces file. The interfaces file is the file, created by SQLEDT (or sybinit on NetWare), which holds the connection information for Open Servers. The default interfaces file is *sql.ini* in the *ini* subdirectory of your *Sybase* directory. On NetWare, the default interfaces file is named *interfaces*.

- S LTM_name** Provides the server name for this LTM. The default LTM_name is DBLTM. The LTM_name must correspond to the Open Server name for the LTM that was entered in SQLEDIT.
- A** Do not filter updates. By default, all changes made by the maintenance user are not replicated. If the **-A** switch is set, these changes are replicated. This may be useful in non-hierarchical Replication Server installations, where a database acts as both a replicate site and as a primary site.
- M** Recovery Mode. This is used to initiate recovery actions. The LTM starts reading logs from the earliest available position. If the offline directory is specified in the configuration file, the LTM reads from the oldest offline log file.
- dl** Display Log Transfer Language (LTL) messages in the LTM window or on the command line and also in the log file. This should be used only to diagnose problems, and is not recommended in a production environment. It carries a significant performance penalty.
- o** Use a log file different from the default (*DBLTM.LOG*). Output messages from log transfer operations are sent to this file for later review.
- ot** Use a log file different from the default (*DBLTM.LOG*), and truncate the log file (all existing content is deleted) when the LTM starts. Output messages from log transfer operations are sent to this file for later review.
- q** Minimize the window when the LTM is started.
- s** Log all LTL commands that are generated by the LTM. This should be used only to diagnose problems, and is not recommended in a production environment. It carries a significant performance penalty.
- v** Displays messages, other than LTL messages, for debugging purposes.

The LTM configuration file

The Adaptive Server Anywhere and Adaptive Server Enterprise LTM configuration files are very similar. This section describes the entries in the Adaptive Server Anywhere LTM configuration file, and the differences from the Adaptive Server Enterprise LTM configuration file.

The configuration file that an LTM uses is specified on the **-C** command-line switch.

LTM configuration
file parameters

The following table describes each of the configuration parameters that the LTM recognizes. Parameters that are used by the Adaptive Server Enterprise LTM but not by the Adaptive Server Anywhere LTM are included in this list, and marked as either ignored (in which case they may be present in the configuration file, but have no effect) or as unsupported (in which case they will cause an error if present in the configuration file).

Parameter	Description
APC_pw	The password for the APC_user login name. This entry is present only in Adaptive Server Anywhere LTM configuration files.
APC_user	A user ID that is used when executing asynchronous procedures at the primary site. This user ID must have permissions appropriate for all asynchronous procedures at the primary site. This entry is present only in Adaptive Server Anywhere LTM configuration files.
backup_only	By default, this is off. If set to on, the LTM will replicate only backed-up transactions
batch_ltl_cmds	Set to on (the default) to use batch mode (which can increase overall throughput significantly).
batch_ltl_sz	The number of commands that are saved in the buffer before being sent to Replication Server, when batch_ltl_cmds is on . The default is 200.
batch_ltl_mem	The amount of memory that the buffer can use before its contents are sent to Replication Server, when batch_ltl_cmds is on . The default is 256K.
continuous	By default, this is on. When set to off, the LTM automatically shuts down as soon as all committed data has been replicated.
LTM_admin_pw	The password for the LTM_admin_user login name.
LTM_admin_user	The system administrator LTM login name that is used to log in to the LTM. This parameter is required so that the LTM can check whether a user logging on to the LTM to shut it down has the correct login name.
LTM_charset	The Open Client/Open Server character set for the LTM to use.
LTM_language	The Open Client/Open Server language for the LTM to use.
LTM_sortorder	The Open Client/Open Server sort order for the LTM to use to compare user names. You can

Parameter	Description
	specify any Adaptive Server Enterprise-supported sort order that is compatible with the LTM's character set. All sort orders in your replication system should be the same.
	The default sort order is a binary sort.
maint_cmds_to_skip	[ignored]
rep_func	Set to on to use replicate function APCs. The default is off .
retry	The number of seconds to wait before retrying a failed connection to an Adaptive Server Anywhere data server or Replication Server. The default is 10 seconds.
RS	The name of the Replication Server to which the LTM is transferring the log.
RS_pwd	The password for the RS_user login name.
RS_source_db	The name of the database whose log the LTM transfers to the Replication Server. This name must match the name of the database as defined within the Replication Server connection definitions. Most configurations use the same setting for both RS_Source_db and SQL_database configuration options.
RS_source_ds	The name of the server whose log the LTM transfers to the Replication Server. This name must match the name of the server as defined within the Replication Server connection definitions. Most configurations use the same setting for both RS_Source_ds and SQL_server configuration options.
RS_user	A login name for the LTM to use to log into Replication Server. The login name must have been granted connect source permission in the Replication Server.
scan_retry	The number of seconds that the LTM waits between scans of the transaction log. This parameter is somewhat different in meaning to that of the Adaptive Server Enterprise LTM. The Adaptive Server Anywhere server does not "wake up" and scan the log when records arrive in the log. For this reason, you may wish to set the <i>scan_retry</i> value to a smaller number than that for an Adaptive Server Enterprise LTM.
skip_ltl_cmd_err	[ignored]

Parameter	Description
SQL_database	The primary site database name on the server SQL_server to which the LTM connects. For Adaptive Server Enterprise during recovery, this is the temporary database whose logs the LTM will transfer to Replication Server. The Adaptive Server Anywhere LTM uses the SQL_log_files parameter to locate offline transaction logs.
SQL_log_files	A directory that holds off-line transaction logs. The directory must exist when the LTM starts up. This entry is present only in Adaptive Server Anywhere LTM configuration files.
SQL_pw	The password for the SQL_user user ID.
SQL_server	The name of the primary site Adaptive Server Anywhere server to which the LTM connects. For Adaptive Server Enterprise during recovery, this is a data server with a temporary database whose logs the LTM will transfer to Replication Server. The LTM uses the SQL_log_files parameter to locate offline transaction logs.
SQL_user	The login name that the LTM uses to connect to the database specified by RS_source_ds and RS_source_db.

Example configuration file

- ◆ The following is a sample LTM configuration file.

```
# This is a comment line
# Names are case sensitive.
SQL_user=sa
SQL_pw=sysadmin
SQL_server=PRIMESV
SQL_database=primedb
RS_source_ds=PRIMEOS
RS_source_db=primedb
RS=MY_REPSERVER
RS_user=sa
RS_pw=sysadmin
LTM_admin_user=DBA
LTM_admin_pw=SQL
LTM_charset=cp850
scan_retry=2
SQL_log_files=e:\logs\backup
APC_user=sa
APC_pw=sysadmin
```

The Log Translation utility

With the Log Translation utility, you can translate a transaction log into a SQL command file.

You can access the Log Translation utility in the following ways:

- ◆ From Sybase Central, for interactive use under Windows 95 or NT.
- ◆ From the system command line, using the *dbtran* command-line utility. This is useful for incorporating into batch or command files.

Translating a transaction log in Sybase Central

❖ **To translate a transaction log into a command file:**

- 1 Open the Database Utilities folder in the left panel.
- 2 Double-click Translate Log in the right panel. The Translate Log wizard is displayed
- 3 Follow the instructions in the wizard.

The DBTRAN command-line utility

Syntax

Running against a transaction log.

dbtran [*switches*] *transaction-log* [*sql-file*]]

Running against a database server.

dbtran [*switches*]

**Windows 3.x
syntax**

Running against a transaction log.

dbtranw [*switches*] *transaction-log* [*sql-file*]]

Running against a database server.

dbtranw [*switches*]

Switch	Description
-a	Include uncommitted transactions
-c " <i>keyword=value; ...</i> "	Supply database connection parameters. Cannot be used with a transaction log name.
-f	Output only since the last checkpoint
-g	Include audit records in output

Switch	Description
-j <i>date/time</i>	Output from the last checkpoint prior to the given time
-n <i>filename</i>	Output SQL file, when used against a database server.
-o <i>filename</i>	Log output messages to file
-q	Run quietly, do not print messages
-r	Remove uncommitted transactions (default)
-s	Produce ANSI standard SQL UPDATE transactions
-t	Include trigger-generated transactions in output
-u <i>userid,...</i>	Translate transactions for listed users only
-x <i>userid,...</i>	Exclude transactions for listed users
-y	Replace file without confirmation
-z	Include trigger-generated transactions as comments only
<i>transaction-log</i>	Log file to be translated. Cannot be used together with -c or -n .
<i>sql-file</i>	Output file containing the translated information. For use with <i>transaction-log</i> only.

Description

The *dbtran* utility takes the information in a transaction log and places it as a set of SQL statements and comments into an output file. The utility can be run in the following ways:

- ◆ **Against a database server** Run in this way, the utility is a standard client application. It connects to the database server using the connection string specified following the **-c** option, and places output in a file specified with the **-n** option.

The following command translates log information from the server **asademo** and places the output in a file named *asademo.sql*.

```
dbtran -c "eng=asademo;uid=dba;pwd=sql" -n asademo.sql
```

DBA authority is required to run in this way.

- ◆ **Against a transaction log file** Run in this way, the utility acts directly against a transaction log file. You should protect your transaction log file from general access if you wish to prevent users from having the capability of running this statement.

```
dbtran asademo.log asademo.sql
```

When the *dbtran* utility runs, it displays the earliest log offset in the transaction log. This can be an effective method for determining the order in which multiple log files were generated.

🌀 For more information about the command-line switches, see "Log translation utility options" on page 100

Log translation utility options

Include uncommitted transactions (-a) The transaction log contains any changes made before the most recent COMMIT by any transaction. Changes made after the most recent commit are not present in the transaction log.

Connection string (-c) When running the command-line utility against a database server, this parameter specifies the connection string.

DBA authority is required to run *dbtran*.

🌀 For a description of the connection parameters, see "Connection parameters" on page 46 of the book *Adaptive Server Anywhere User's Guide*.

Translate from last checkpoint only (-f) Only transactions that were completed since the last checkpoint are translated.

Include audit information (-g) If the AUDITING database option is turned on, auditing information is added to the transaction log. You can include this information as comments in the output file using this option.

🌀 For more information, see "AUDITING option" on page 140.

Output from the last checkpoint prior to a given date (-j) Only transactions from the most recent checkpoint prior to the given date and/or time are translated. The user-provided argument can be a date, time or date and time enclosed in quotes. If the time is omitted, the time is assumed to be the beginning of the day. If the date is omitted, the current day is assumed. The following is an acceptable format for the date and time: "YY/MM/DD HH:MM".

Output file (-n) When you run the *dbtran* command-line utility against a database server, use this option to specify the output file that holds the SQL statements.

Output log messages to file (-o) Redirect log messages to the named file.

Operate quietly (-q) Do not display log messages. This option is available only from the command-line utility.

Do not include uncommitted transactions (-r) Remove any transactions that were not committed and the default operation.

Generate ANSI standard SQL UPDATE (-s) If is no primary key or unique index on a table, the Translation utility generates UPDATE statements with a non-standard FIRST keyword in case of duplicate rows. The ANSI standard UPDATE flag does not output this keyword.

Include transactions generated by triggers (-t) By default, actions carried out by triggers are not included in the command file. If the matching trigger is in place in the database, when the command file is run against the database, the trigger will carry out the actions automatically. Trigger actions should be included if the matching trigger does not exist in the database against which the command file is to be run.

Translate transactions for listed users only (-u)

Translate transactions except for listed users (-x)

Operate without confirming actions (-y) Without this option, you are prompted to confirm the replacement of an existing command file.

Include transactions generated by triggers as comments only (-z) Transactions that were generated by triggers will be included only as comments in the output file.

The REBUILD component

Databases can be unloaded and rebuilt using either of two rebuild components: the Rebuild wizard in Sybase Central or the Rebuild batch or command file, which invokes a series of utilities to rebuild a database.

Rebuilding a database using the Rebuild wizard in Sybase Central

Sybase Central's Rebuild wizard lets you change the database configuration while rebuilding a database. The configuration settings that can be changed include page size and some database attributes (encryption, tailing blanks, and case sensitivity).

❖ **To rebuild a database:**

- 1 Open the Database Utilities folder in the left panel.
- 2 Double-click Rebuild Database in the right panel. The Rebuild wizard is displayed.
- 3 Follow the instructions in the wizard.

Rebuilding a database using the REBUILD batch or command file

Syntax `rebuild old-database new-database [dba-password]`

See also "The DBUNLOAD command-line utility" on page 111
"The DBINIT command-line utility" on page 84

Description This Windows NT batch file or UNIX script uses *dbunload* to rebuild *old-database* into *new-database*. This is a simple script, but it helps document and automate the rebuilding process. Both database names should be specified without extensions. An extension of DB will automatically be added.

The *dba-password* must be specified if the password to the DBA user ID in the *old-database* is not the initial password SQL.

REBUILD runs the *dbunload*, *dbinit*, and Interactive SQL commands with the default command line options. If you need different options, you will need to run the three commands separately.

The Stop utility

The Stop utility stops a database server.

The Stop utility is a command-line utility only. In Windowing environments, you can stop a database server by clicking Close on the server window or choosing Exit from the File menu on the server window.

The DBSTOP command-line utility

Syntax `dbstop [switches] { name | CLIENT }`

Windows 3.x syntax `dbstopw [switches] { name | CLIENT }`

Switch	Description
<code>-c "keyword=value; ..."</code>	Connection parameters
<code>-q</code>	Quiet mode—do not print messages
<code>-x</code>	Do not stop if there are active connections
<code>-y</code>	Stop without prompting even if there are active connections

Description

In UNIX, *dbstop* can shut down a server on any node on the network. The *name* is necessary to specify the name of the server that you wish to stop.

Command-line options let you control whether a server is stopped even if there are active connections.

☞ For more information about the command-line switches, see "Stop utility options" on page 103.

Stop utility options

-c When stopping a network server, you must supply a connection string with a user ID that has permissions to stop the server. By default, DBA permission is required on the network server, and all users can shut down a personal server, but the `-gk` server command-line option can be used to change this.

☞ For a description of the connection parameters, see "Connection parameters" on page 46 of the book *Adaptive Server Anywhere User's Guide*.

The behavior of *dbstop* if there are active connections on a server can be controlled. If there are active connections, *dbstop* provides a prompt asking if you wish to shut down the server. If you specify **unconditional=true** on the command line, the server is shut down without prompting, even if there are active connections.

-q Operate quietly. Do not print a message if the database was not running.

-x Do not stop the server if there are still active connections to the server.

-y Stop the server even if there are still active connections to the server.

UNIX switch

-p password Specify a password to unlock the server. If the server's keyboard is locked and no password is specified, or the specified password is incorrect, the server will not shut down.

The Transaction Log utility

With the Transaction Log utility, you can display or change the name of the transaction log or transaction log mirror associated with a database. You can also stop a database from maintaining a transaction log or mirror, or start maintaining a transaction log or mirror.

A transaction log mirror is a duplicate copy of a transaction log, maintained by the database in tandem.

The name of the transaction log is first set when the database is initialized. The Transaction Log utility works with database files or with write files. The database server must not be running on that database when the transaction log filename is changed (or an error message is displayed).


You can access the Transaction Log utility in the following ways:

- ◆ From Sybase Central, for interactive use under Windows 95 or NT.
- ◆ From the system command line, using the *dblog* command-line utility.

Changing a log file name from Sybase Central

❖ **To change a transaction log file name:**

- 1 Open the Database Utilities folder in the left panel.
- 2 Double-click Change Log File in the right panel. The Transaction Log wizard is displayed
- 3 Follow the instructions in the wizard.

 For full information on changing a log file name from Sybase Central, see the Sybase Central online Help.


The DBLOG command-line utility

Syntax	dblog [<i>switches</i>] <i>database-file</i>
Windows 3.x syntax	dblogw [<i>switches</i>] <i>database-file</i>

Switch	Description
-g <i>n</i>	Sets the generation number to <i>n</i>
-il	Ignores the LTM truncation offset stored in the database
-ir	Ignores the SQL Remote truncation offset stored in the database
-m <i>mirror-name</i>	Set transaction log mirror name
-n	No longer use a transaction log
-o <i>file</i>	Output messages to file
-q	Quiet mode—do not print messages
-r	No longer use a transaction log mirror
-t <i>log-name</i>	Set the transaction log name
-x <i>n</i>	Set the transaction log current relative offset to <i>n</i>
-z <i>n</i>	Set the transaction log starting offset to <i>n</i>

Description

The DBLOG command line utility allows you to display or change the name of the transaction log or transaction log mirror associated with a database. You can also stop a database from maintaining a transaction log or mirror, or start maintaining a transaction log or mirror.

 For more information about the command-line switches, see "Transaction log utility options" on page 106.

Transaction log utility options

Set the generation number (-g) Use this option if you are using the Log Transfer Manager to participate in a Replication Server installation. It can be used after a backup is restored, to set the generation number. It performs the same function as the following Replication Server function:

```
dbcc settrunc( 'ltm', 'gen_id', n ).
```

For information on generation numbers and dbcc, see your Replication Server documentation.

Ignore the LTM truncation offset (-il) Use this option if you are using the Log Transfer Manager to participate in a Replication Server installation. It performs the same function as the following Replication Server function:

```
dbcc settrunc( 'ltm', 'gen_id', n ).
```

For information on dbcc, see your Replication Server documentation.

Ignore the SQL Remote truncation offset (-ir) Use this option if you are using the Log Transfer Manager to participate in a Replication Server installation and a SQL Remote installation. It resets the offset that is kept for the DELETE_OLD_LOGS option, allowing transaction logs to be deleted when they are no longer needed.

Set the name of the transaction log mirror file (-m) This option sets a filename for a new transaction log mirror. If the database is not currently using a transaction log mirror, it starts using one. If the database is already using a transaction log mirror, it changes to using the new file as its transaction log mirror.

No longer use a transaction log (-n) Stop using a transaction log. Without a transaction log, the database will no longer be able to participate in data replication or use the transaction log in data recovery.

Output log messages to file (-o) Redirect log messages to the named file.

Operate quietly (-q) Do not display messages on a window. This option is available only from the command-line utility.

No longer use a transaction log mirror (-r) For databases that maintain a mirrored transaction log, this option changes their behavior to maintain only a single transaction log.

Set the name of the transaction log file (-t) This option sets a filename for a new transaction log. If the database is not currently using a transaction log, it starts using one. If the database is already using a transaction log, it changes to using the new file as its transaction log.

Set the current log offset (-x) For use when reloading a SQL Remote consolidated database. This option resets the current log offset so that the database can take part in replication.

☞ For information on how to use this option, see "Unloading and reloading a consolidated database" on page 272 of the book *Data Replication with SQL Remote*.

Set the starting log offset (-z) For use when reloading a SQL Remote consolidated database. This option resets the starting log offset so that the database can take part in replication.

☞ For information on how to use this option, see "Unloading and reloading a consolidated database" on page 272 of the book *Data Replication with SQL Remote*.

The Uncompression utility

With the Uncompression utility, you can expand a compressed database file created by the Compression utility. The Uncompression utility reads the compressed file and creates an uncompressed database file.

The Uncompression utility does not uncompress files other than the main database file (dbspace files).


You can access the Uncompression utility in the following ways:

- ◆ From Sybase Central, for interactive use under Windows 95 or NT.
- ◆ From the system command line, using the *dbexpand* command-line utility. This is useful for incorporating into batch or command files.

Uncompressing a database in Sybase Central

❖ **To uncompress a compressed database file:**

- 1 Open the Database Utilities folder in the left panel.
- 2 Double-click Uncompress Database in the right panel. The Uncompress wizard is displayed.
- 3 Follow the instructions in the wizard.

 For full information on uncompressing a database in Sybase Central, see the Sybase Central online Help.

The DBEXPAND command-line utility

Syntax

dbexpand [*switches*] *compressed-database-file* [*database-file*]

**Windows 3.x
syntax**

dbexpandw [*switches*] *compressed-database-file* [*database-file*]

Switch	Description
-o	Output log messages to a file
-q	Operate quietly—do not print messages
-y	Erase an existing output file without confirmation

Description

The input filename extension defaults to *cdb*. The output filename (with extension) must not have the same name as the input filename (with extension).

☞ For more information about the command-line switches, see "Uncompression utility options" on page 109.

Uncompression utility options

Output log messages to file (-o) Redirect log messages to the named file.

Operate quietly (-q) Do not display messages on a window. This option is available only from the command-line utility.

Operate without confirming actions (-y) Without this option, you are prompted to confirm the replacement of an existing database file.

The Unload utility

With the Unload utility, you can unload a database and put a set of data files in a named directory. The Unload utility creates the Interactive SQL command file *reload.sql* to rebuild your database from scratch. It also unloads all of the data in each of your tables, into files in the specified directory in comma-delimited format. Binary data is properly represented with escape sequences.

You can also use the Unload utility to directly create a new database from an existing one. This avoids potential security problems with the database contents being written to disk.

Accessing the Unload utility

You can access the Unload utility in the following ways:

- ◆ From Sybase Central, for interactive use under Windows 95 or NT.
- ◆ From the system command line, using the *dbunload* command-line utility. This is useful for incorporating into batch or command files.

The Unload utility should be run from a user ID with DBA authority. This is the only way you can be sure of having the necessary privileges to unload all the data. In addition, the *reload.sql* file should be run from the DBA user ID. (Usually, it will be run on a new database where the only user ID is **DBA** with password **SQL**.)

Objects owned by dbo

The **dbo** user ID owns a set of Adaptive Server Enterprise-compatible system objects in a database.

The Unload utility does not unload the objects that were created for the **dbo** user ID during database creation. Changes made to these objects, such as redefining a system procedure, are lost when the data is unloaded. Any objects that were created by the **dbo** user ID since the initialization of the database are unloaded by the Unload utility, and so these objects are preserved.

Unloading and replication

There are special considerations for unloading databases involved in replication.

🔗 For information, see "Unloading and reloading a consolidated database" on page 272 of the book *Data Replication with SQL Remote*.

Unloading a database from Sybase Central

❖ To unload a running database:

- 1 Connect to the database.

- 2 Right-click the database and click Unload in the popup menu. The Unload Database wizard is displayed.
- 3 Follow the instructions in the wizard.

❖ **To unload a database file or a running database:**

- 1 Open the Database Utilities folder in the left panel.
- 2 Double-click Unload Database in the right panel. The Unload Database Wizard is displayed.
- 3 Follow the instructions in the Wizard.

☞ For full information on unloading a database from Sybase Central, see the Sybase Central online Help.

The DBUNLOAD command-line utility

Syntax

dbunload [*switches*] [*directory*]

Windows 3.x syntax

dbunloaw [*switches*] [*directory*]

Switch	Description
-ac " <i>keyword=value; ...</i> "	Connect to the database specified in the connect string to do the reload.
-an <i>database</i>	Creates a database file with the same settings as the database being unloaded, and automatically reloads it.
-c " <i>keyword=value; ...</i> "	Supply database connection parameters
-d	Unload data only
-e <i>list</i>	No data output for listed tables
-ii	Internal unload, internal reload (default)
-ix	Internal unload, external reload
-j <i>nnn</i>	Repeated unload of view creation statements
-n	No data—schema definition only
-o <i>file</i>	Log output messages to <i>file</i>
-p <i>char</i>	Escape character for external unloads (default "\\")
-q	Quiet mode—no windows or messages
-r <i>reload-file</i>	Specify name and directory of generated reload Interactive SQL command file (default <i>reload.sql</i>)
-t <i>list</i>	Output only the listed tables

Switch	Description
-u	Unordered data
-v	Verbose messages
-xi	External unload (when client and server are on different machines), internal reload
-xx	External unload (when client and server are on different machines), external reload
-y	Replace the command file without confirmation
directory	The directory where the unloaded data is to be placed. Not needed for -an or -ac .

Description

In the default mode, the directory used by *dbunload* to hold the data is relative to the database server, not to the current directory of the user. For details of how to supply a filename and path in this mode, see "UNLOAD TABLE statement" on page 570.

If the **-x** switch is used, the directory is relative to the current directory of the user. The *reload.sql* command file is always relative to the current directory of the user, regardless of whether **-x** is used.

If no list of tables is supplied, the whole database is unloaded. If a list of tables is supplied, only those tables are unloaded.

☞ For more information about the command-line switches, see "Unload utility options" on page 112.

☞ There are special considerations for unloading databases involved in replication. For information, see "Unloading and reloading a consolidated database" on page 272 of the book *Data Replication with SQL Remote*.

Unload utility options

Reload the data to an existing database (-ac) You can combine the operation of unloading a database and reloading the results into an existing database using this option.

For example, the following command (which should be entered all on one line) loads a copy of the *asademo.db* database into an existing database file named *newdemo.db*:

```
dbunload -c "uid=dba;pwd=sql;dbf=asademo.db" -ac  
"uid=dba;pwd=sql;dbf=newdemo.db"
```

If you use this option, no interim copy of the data is created on disk, so you do not specify an unload directory on the command line. This provides greater security for your data, but at some cost for performance.

Create a database for reloading (-an) You can combine the operations of unloading a database, creating a new database, and loading the data using this option.

For example, the following command (which should be entered all on one line) creates a new database file named *asacopy.db* and copies the schema and data of *asademo.db* into it:

```
dbunload -c "uid=dba;pwd=sql;dbf=asademo.db" -an
asacopy.db
```

If you use this option, no interim copy of the data is created on disk, so you do not specify an unload directory on the command line. This provides greater security for your data, but at some cost for performance.

Connection parameters (-c) For a description of the connection parameters, see "Connection parameters" on page 46 of the book *Adaptive Server Anywhere User's Guide*. If the connection parameters are not specified, connection parameters from the SQLCONNECT environment variable are used, if set. The **user ID** should have DBA authority, to ensure that the user has permissions on all the tables in the database.

For example, the following statement unloads the **asademo** database running on the **sample_server** server, connecting as user ID **DBA** with password **SQL**. The data is unloaded into the *c:\unload* directory.

```
dbunload -c
"eng=sample_server;dbn=asademo;uid=dba;pwd=sql"
c:\unload
```

Unload data only (-d) With this option, none of the database definition commands are generated (CREATE TABLE, CREATE INDEX, and so on); *reload.sql* contains statements to reload the data only.

No data output for listed tables (-e) This is accessible only from the command-line utility. By default, the optional table-list defines the tables to be unloaded. If you wish to unload almost all of the tables in the database, the **-e** command-line switch unloads all tables except the specified tables.

Internal versus external unloads and reloads

The following switches offer combinations of internal and external unloads and reloads: `-ii`, `-ix`, `-xi` and `-xx`. A significant performance gain can be realized by using internal commands (UNLOAD/LOAD) versus external commands (Interactive SQL's INPUT and OUTPUT statement). However, internal commands are executed by the server, so file and directory paths are relative to the location of the database server. Using external commands, file and directory paths are relative to the current directory of the user.

Use internal unload, internal reload (-ii) Accessible only from the command-line utility, this switch uses the UNLOAD statement to extract data from the database, and uses the LOAD statement in the *reload.sql* file to repopulate the database with data. This is the default.

Use internal unload, external reload (-ix) Accessible only from the command-line utility, this switch uses the UNLOAD statement to extract data from the database, and uses the Interactive SQL INPUT statement in the *reload.sql* file to repopulate the database with data.

Repeated unload of view creation statements (-j) If your database contains view definitions that are dependent on each other, you can use this option to avoid failure when reloading the views into a database. This option causes view creation statements to be unloaded multiple times, as specified by the count entered. This count should be small, and should correspond to the number of levels of view dependency.

Unload schema definition only (-n) With this option, none of the data in the database is unloaded; *reload.sql* contains SQL statements to build the structure of the database only.

Log output messages to a file (-o) Logs all output messages to a file.

Escape character (-p) The default escape character (`\`) for external unloads (DBUNLOAD `-x` switch) can be replaced by another character, using this option. This option is available only from the command-line utility.

Operate quietly (-q) Display no messages except errors. This option is available only from the command-line utility.

Specify reload filename (-r) Modify the name and directory of the generated reload Interactive SQL command file. The default is *reload.sql* in the current directory.


Unload only listed tables (-t) Provide a list of tables to be unloaded. By default, all tables are unloaded. Together with the `-n` option, this allows you to unload a set of table definitions only.

Output the data unordered (-u) Normally, the data in each table is ordered by the primary key. Use this option if you are unloading a database with a corrupt index, so that the corrupt index is not used to order the data.

Enable verbose mode (-v) The table name of the table currently being unloaded, and how many rows have been unloaded, is displayed. This option is available only from the command-line utility.

Use external unloading, internal reload (-xi) Accessible only from the command-line utility, this switch uses the Interactive SQL OUTPUT statement to extract data from the database, and uses the LOAD statement in the generated reload command file, *reload.sql*, to repopulate the database with data.

Use external unloading, external reload (-xx) Accessible only from the command-line utility, this switch uses the Interactive SQL OUTPUT statement to extract data from the database, and uses the Interactive SQL INPUT statement in the generated reload command file, *reload.sql*, to repopulate the database with data.

 For more information on filenames and paths for the Unload utility, see "UNLOAD TABLE statement" on page 570.

Operate without confirming actions (-y) Without this option, you are prompted to confirm the replacement of an existing command file.

Rebuilding a database

To unload a database, start the database server with your database, and run the Unload utility with the DBA user ID and password.

To reload a database, create a new database and then run the generated *reload.sql* command file through Interactive SQL.

In Windows 95 and NT, and UNIX, there is a file (*rebuild.bat*, *rebuild.cmd*, or *rebuild*) that automates the unload and reload process.

The Upgrade utility

The Upgrade utility upgrades a database created with Watcom SQL 3.2, Watcom SQL 4.0, or SQL Anywhere 5.0 to the Adaptive Server Anywhere format. While Adaptive Server Anywhere does run against databases that were created with earlier releases of the software, some of the features that were introduced since the version that created the database are unavailable unless the database is upgraded.

For people switching from release 3.2 or 4.0 to the current release, the Upgrade utility upgrades their databases without unloading and reloading them.

If you wish to use replication on an upgraded database, you must also archive your transaction log and start a new one on the upgraded database.

You can access the Upgrade utility in the following ways:

- ◆ From Sybase Central, for interactive use under Windows 95 or NT.
- ◆ From the system command-line, using the *dbupgrad* command-line utility.

Upgrading a database from Sybase Central

❖ **To upgrade a database:**

- 1 Open the Database Utilities folder in the left panel.
- 2 Double-click Upgrade Database in the right panel. The Upgrade Database wizard is displayed.
- 3 Follow the instructions in the wizard.

🔗 For full information on upgrading a database from Sybase Central, see the Sybase Central online Help.

Upgrading databases that are too old for the Upgrade utility

❖ **To upgrade a database created with a version of Adaptive Server Anywhere that is too old to be upgraded:**

- 1 Unload the database using the Unload utility.
- 2 Create a database with the name you that you wish to use for the upgraded version, using the Initialization utility.

- 3 Connect to the new database from Interactive SQL as the DBA user ID, and read the reload.sql command file to build the new database.

The DBUPGRAD command-line utility


Syntax `dbupgrad [switches]`

Windows 3.x syntax `dbupgrdw [switches]`

Switch	Description
-c "keyword=value; ..."	Supply database connection parameters
-i	Do not install Sybase jConnect support
-j	Do not install Sybase runtime Java classes
-k	Close the window on completion
-q	Quiet mode—no windows or messages

Description

The DBUPGRAD command-line utility upgrades a database created with Watcom SQL 3.2, Watcom SQL 4.0, or SQL Anywhere 5.0 to the Adaptive Server Anywhere format. While Adaptive Server Anywhere does run against databases that were created with earlier releases of the software, some of the features that were introduced since the version that created the database are unavailable unless the database is upgraded.

 For more information about the command-line switches, see "Upgrade utility options" on page 117.

Upgrade utility options

Connection parameters (-c) For a description of the connection parameters, see "Connection parameters" on page 46 of the book *Adaptive Server Anywhere User's Guide*. If the connection parameters are not specified, connection parameters from the SQLCONNECT environment variable are used, if set. The **user ID** must have DBA authority.

For example, the following command upgrades a database called sample40 to a 6.0 format, connecting as user **DBA** with password **SQL**:

```
dbupgrad -c "uid=dba;pwd=sql;dbf=c:\wsq140\sample40.db"
```

The *dbupgrad* command-line utility must be run by a user with DBA authority.

Do not install Sybase jConnect support (-i) If you wish to use the Sybase jConnect JDBC driver to access system catalog information, you need to install jConnect support. Use this option if you wish to exclude the jConnect system objects. You can still use JDBC, as long as you do not access system information.

Do not install Sybase runtime Java classes (-j) If you wish to use Java in your database, you must install the Sybase runtime Java classes. By default, these classes are installed when a database is upgraded to Version 6. The runtime classes add several megabytes to the size of a database. If you do not intend to use Java classes, you can specify the `-j` switch to avoid installing these classes.

Close the window on completion (-k) Once the upgrade is completed, the message window is closed. This option is available only from the command-line utility.

Operate quietly (-q) Do not display messages on a window. This option is available only from the command-line utility.

The Validation utility

With the Validation utility, you can validate the indexes and keys on some or all of the tables in the database. The Validation utility scans the entire table, and looks up each record in every index and key defined on the table.

This utility can be used in combination with regular backups (see "Backup and Data Recovery" on page 553 of the book *Adaptive Server Anywhere User's Guide*) to give you confidence in the integrity of the data in your database.

You can access the Validation utility in the following ways:

- ◆ From Sybase Central, for interactive use under Windows 95 or NT.
- ◆ From the system command line, using the *dbvalid* command-line utility. This is useful for incorporating into batch or command files.

☞ For more information on validating tables, see "VALIDATE TABLE statement" on page 577.

Validating a database from Sybase Central

❖ To validate a running database:

- 1 Connect to the database.
- 2 Right-click the database, and click Validate in the popup menu.

❖ To validate an individual table:

- 1 Connect to the database.
- 2 Locate the table you wish to validate.
- 3 Right-click the table, and click Validate in the popup menu.

☞ For full information on validating a database from Sybase Central, see the Sybase Central online Help.


The DBVALID command-line utility

Syntax	dbvalid [<i>switches</i>] [<i>table-name,...</i>]
Windows 3.x syntax	dbvalidw [<i>switches</i>] [<i>table-name,...</i>]

Switch	Description
-c "keyword=value; ..."	Supply database connection parameters
-q	Quiet mode—do not print messages

Description

With the command line Validation utility, you can validate the indexes and keys on some or all of the tables in the database. This utility scans the entire table, and looks up each record in every index and key defined on the table.

 For more information about the command-line switches, see "Validation utility options" on page 120.

Validation utility options

Connection parameters (-c) For a description of the connection parameters, see "Connection parameters" on page 46 of the book *Adaptive Server Anywhere User's Guide*. If the connection parameters are not specified, connection parameters from the SQLCONNECT environment variable are used, if set. The user ID must have DBA authority.

For example, the following validates the sample database, connecting as user **DBA** with password **SQL**:

```
dbvalid -c "uid=dba;pwd=sql;dbf=c:\asa6\asdemo.db"
```

Operate quietly (-q) Do not display messages on a window. This option is available only from the command-line utility.

The Write File utility

The Write File utility is used to manage database write files. A **write file** is a file attached to a particular database. All changes are written into the write file, leaving the database file unchanged.

Using write files for development

Write files can be used effectively for testing when you do not wish to modify the production database. They can also be used in network and student environments where read-only access to a database is desired, or when you distribute on CD-ROM a database that you wish users to be able to modify.

Compressed databases

If you are using a compressed database, you must use a write file; compressed database files cannot be modified directly. The write file name is then used in place of the database name when connecting to the database, or when loading a database on the database server command line.

❖ **You can access the write file utility in the following ways:**


- ◆ From Sybase Central, for interactive use under Windows 95 or NT.
- ◆ From the system command line, using the *dbwrite* command-line utility. This is useful for incorporating into batch or command files.

The Write File utility runs against a database file. The database must not be running on a server when you run the Write File utility.

Creating a write file from Sybase Central

❖ **To create a write file for a database:**

- 1 Open the Database Utilities folder in the left panel.
- 2 Double-click Create Write File in the right panel. The Write File wizard is displayed.
- 3 Follow the instructions in the wizard.

 For full information on creating a write file from Sybase Central, see the Sybase Central online Help.

The DBWRITE command-line utility

Syntax

dbwrite [*switches*] *database-file* [*write-name*]

**Windows 3.x
syntax**

dbwritew [*switches*] *database-file* [*write-name*]

Switch	Description
-c	Create a new write file
-d <i>database-file</i>	Point a write file to a different database
-f <i>database-file</i>	Force the write file to point at a file
-m <i>mirror-name</i>	Set the transaction log mirror name
-o <i>file</i>	Output messages to file
-q	Quiet mode—do not print messages
-s	Report the write file status only
-t <i>log-name</i>	Set the transaction log name
-y	Erase/replace old files without confirmation

Description

If any changes are made to the original database (not using the write file), the write file will no longer be valid. This happens if you start the server using the original database file and make a modification to it. If a write file becomes invalid, you can discard all of your changes and create a new write file with the following command.

```
dbwrite -c db-name write-name
```

The *log-name* and *mirror-name* parameters are used only when creating a new write file. The *write-name* parameter is used only with the -c and -d parameters. Note that the *database_file* parameter must be specified before the *write-name* parameter.

☞ For more information about the command-line switches, see "Write file utility options" on page 122.

Write file utility options

Create a new write file (-c) If an existing write already exists, any information in the old write file will be lost. If no write file name is specified on the command line, the write filename defaults to the database name with the extension *wrt*. If no transaction log name is specified, the log file name will default to the database name with the extension *wlg*.

Change the database file to which an existing write file points (-d) If a database file is moved to another directory, or renamed, this option allows you to maintain the link between the write file and the database file. This option is available only from the command-line utility.

Force a write file to point to a file (-f) This option is available only from the command-line utility. This option is for use when a write file is being created and the database file is held on a Novell NetWare or other network path, for operating systems on which they cannot be entered directly. By providing the full Novell path name for the database file (for example: *SYS\asademo.db*), you can avoid dependencies on local mappings of the NetWare path. Unlike with the `-d` option, no checking is done on the specified path.

Operate quietly (-q) Do not display messages on a window. This option is available only from the command-line utility.

Report the write file status only (-s) This displays the name of the database to which the write file points. This option is available only from the command-line utility.

Operate without confirming actions (-y) Without this option, you are prompted to confirm the replacement of an existing write file.

The SQL preprocessor

The SQL preprocessor processes a C or C++ program with embedded SQL, before the compiler is run.

Syntax

sqlpp [*switches*] *sql-filename* [*output-filename*]

Switch	Description
-c "keyword=value;..."	Supply database connection parameters [Ultra:Lite]
-d	Favor data size
-e <i>level</i>	Flag non-conforming SQL syntax as an error
-f	Put the far keyword on generated static data
-h <i>line-width</i>	Limit the maximum line length of output
-n	Line numbers
-o <i>operating-sys</i>	Target operating system: WINNT, WINDOWS, NETWARE, UNIX, UNIX64, OS232
-p	Project name [UltraLite]
-q	Quiet mode—do not print banner
-r	Generate reentrant code
-s <i>string-len</i>	Maximum string length for the compiler
-w <i>level</i>	Flag non-conforming SQL syntax as a warning
-x	Change multibyte SQL strings to escape sequences.
-z <i>sequence</i>	Specify collation sequence

See also

"Application development using Embedded SQL" on page 8 of the book *Adaptive Server Anywhere Programming Interfaces Guide*

Description

The SQL preprocessor processes a C or C++ program with embedded SQL before the compiler is run. SQLPP translates the SQL statements in the *input-file* into C language source that is put into the *output-file*. The normal extension for source programs with embedded SQL is SQC. The default output filename is the **sql-filename** with an extension of C. If the **sql-filename** has a C. extension, the output filename extension is CC by default.

Switches

- c Reserved for future use by UltraLite. Not supported in this release.
- d Generate code that reduces data space size. Data structures will be reused and initialized at execution time before use. This increases code size.

-e level This option flags any Embedded SQL that is not part of a specified set of SQL/92 as an error.

The allowed values of *level* and their meanings are as follows:

- ◆ **e** flag syntax that is not entry-level SQL/92 syntax
- ◆ **i** flag syntax that is not intermediate-level SQL/92 syntax
- ◆ **f** flag syntax that is not full-SQL/92 syntax
- ◆ **t** flag non-standard host variable types
- ◆ **w** allow all supported syntax

-f Put the *far* keyword in front of preprocessor-generated data. This may be required in conjunction with the Borland C++ compiler for the large memory model. By default, all static data is put in the same segment. Adding the *far* keyword will force static data into different segments. (By default, WATCOM C and Microsoft C place data objects bigger than a threshold size in their own segment.)

-h num Limits the maximum length of lines output by *sqlpp* to *num*. The continuation character is a backslash (\), and the minimum value of *num* is ten.

-n Generate line number information in the C file. This consists of *#line* directives in the appropriate places in the generated C code. If the compiler that you are using supports the *#line* directive, this switch will make the compiler report errors on line numbers in the SQC file (the one with the Embedded SQL) as opposed to reporting errors on line numbers in the C file generated by the SQL preprocessor. Also, the *#line* directives will indirectly be used by the source level debugger so that you can debug while viewing the SQC source file.

-o operating-sys Specify the target operating system. Note that this option must match the operating system where you will run the program. A reference to a special symbol will be generated in your program. This symbol is defined in the interface library. If you use the wrong operating system specification or the wrong library, an error will be detected by the linker. The supported operating systems are:

- ◆ **WINDOWS** Microsoft Windows 3.x
- ◆ **WIN32** 32-bit Microsoft Windows
- ◆ **WINNT** Microsoft Windows NT
- ◆ **NETWARE** Novell NetWare
- ◆ **UNIX** UNIX

◆ **UNIX 64** 64-bit UNIX

-p Reserved for future use by UltraLite. Not supported in this release.

-q Operate quietly. Do not print the banner.

-r Generate reentrant code (See "SQLCA management for multi-threaded or reentrant code" on page 29 of the book *Adaptive Server Anywhere Programming Interfaces Guide*).

-s string-len Set the maximum size string that the preprocessor will put into the C file. Strings longer than this value will be initialized using a list of characters ('a','b','c', etc). Most C compilers have a limit on the size of string literal they can handle. This option is used to set that upper limit. The default value is 500.

-w level This option flags any Embedded SQL that is not part of a specified set of SQL/92 as a warning.

The allowed values of *level* and their meanings are as follows:

◆ **e** flag syntax that is not entry-level SQL/92 syntax

◆ **i** flag syntax that is not intermediate-level SQL/92 syntax

◆ **f** flag syntax that is not full-SQL/92 syntax

◆ **t** flag non-standard host variable types

◆ **w** allow all supported syntax

-x Change multibyte strings to escape sequences so that they can pass through compilers.

-z sequence This option specifies the collation sequence or filename. (For a listing of available collation sequences, type **dbinit -l** at the command prompt).