CHAPTER 5

Database Options

About this chapter

This chapter describes the database and Interactive SQL options that can be set to customize and modify database behavior.

Contents

Introduction to database options

Database options control many aspects of database behavior. For example, you can use database options for the purposes such as the following:

- **Compatibility** You can control how much like Adaptive Server Enterprise your Adaptive Server Anywhere database operates, and whether SQL that does not conform to SQL/92 generates errors.
- **Error handling** You can control what happens when errors such as dividing by zero, or overflow errors, occur.
- **Concurrency and transactions** You can control the degree of concurrency, and details of COMMIT behavior, using options.

Setting options

You set options with the SET OPTION statement. It has the following general syntax:

SET [**TEMPORARY**] **OPTION**
... [*userid*. | **PUBLIC**.]*option-name* = [*option-value*]

Specify a user ID or group name to set the option for that user or group only. Every user belongs to the PUBLIC group. If no user ID or group is specified, the option change is applied to the currently logged on user ID that issued the SET OPTION statement.

For example, the following statement applies an option change to the user DBA, if DBA is the user that issues it:

```
SET OPTION login_mode = mixed
```

The following statement applies a change to the **PUBLIC** user ID, a user group to which all users belong.

```
SET OPTION Public.login_mode = standard
```

Finding option settings

Current option settings for your connection are available as connection properties. You can list all connection properties using the **sa_conn_properties** system procedure.

```
call sa_conn_properties
```

You can obtain a single setting using the **connection_property** system function. For example, the following statement reports the value of the **Ansi_integer_overflow** option:

```
SELECT connection_property ('Ansi_integer_overflow')
```

The scope of database options

Some options (such as COMMIT behavior) are database-wide in scope. Setting these options requires DBA permissions. Other options apply only to the current connection (such as ISOLATION_LEVEL), and need no special permissions.

Setting public options

Only users with DBA privileges have the authority to set an option for the **PUBLIC** user ID.

Changing the value of an option for the **PUBLIC** user ID sets the value of the option for all users who have not SET their own value. An option value cannot be set for an individual user ID unless there is already a **PUBLIC** user ID setting for that option.

Setting temporary options

Adding the TEMPORARY keyword to the Set Option statement changes the duration of the change. Ordinarily an option change is permanent: it will not change until it is explicitly changed using the Set Option statement.

When the SET TEMPORARY OPTION statement is applied using an individual user ID, the new option value is in effect as long as that user is logged in to the database.

When the SET TEMPORARY OPTION is used with the **PUBLIC** user ID, the change is in place for as long as the database is running. When the database is shut down, Temporary options for the **PUBLIC** user ID revert back to their permanent value.

Setting an option for the **PUBLIC** user ID temporarily offers a security advantage. For example, when the login_mode option is enabled the database relies on the login security of the system on which it is running. Enabling it temporarily means that a database relying on the security of a Windows NT domain will not be compromised if the database is shut down and copied to a local machine. In this case, the login_mode option will revert to its permanent value, which could be Standard, a mode where integrated logins are not permitted.

Deleting option settings

If *option-value* is omitted, the specified option setting will be deleted from the database. If it was a personal option setting, the value reverts back to the PUBLIC setting. If a TEMPORARY option is deleted, the option setting reverts back to the permanent setting.

> ***Caution***
> *Changing option settings while fetching rows from a cursor is not supported, because it can lead to ill-defined behavior. For example, changing the DATE_FORMAT setting while fetching from a cursor would lead to different date formats among the rows in the result set. Do not change option settings while fetching rows.*

Option classification

Adaptive Server Anywhere provides many options. It is convenient to divide them into a few general classes. The classes of options are:

- General database options
- Transact-SQL compatibility database options
- Replication database options
- Interactive SQL options

Initial option settings

Connections to Adaptive Server Anywhere can be made through the TDS protocol (Open Client and jConnect JDBC connections) or through the Adaptive Server Anywhere protocol (ODBC, Embedded SQL).

If you have users who use both TDS and the Adaptive Server Anywhere-specific protocol, you can configure their initial settings using stored procedures. As it is shipped, Adaptive Server Anywhere uses this method to set Open Client connections and jConnect connections to reflect default Adaptive Server Enterprise behavior.

The initial settings are controlled using the LOGIN_PROCEDURE option. This option names a stored procedure to run when users connect. The default setting is to use the **sp_login_environment** system procedure. If you wish to change this behavior you can do so.

In its turn, **sp_login_environment** checks to see if the connection is being made over TDS. If it is, it calls the **sp_tsql_environment** procedure, which sets several options to new "default" values for the current connection.

For more information, see "LOGIN_PROCEDURE option" on page 161, "sp_login_environment system procedure" on page 757, and "sp_tsql_environment system procedure" on page 759.

Duration of options

Changes to option settings take place at different times, depending on the option. Changing a global option such as RECOVERY_TIME takes place the next time the server is started.

Options that affect the current connection only generally take place immediately. You can change option settings in the middle of a transaction, for example. One exception to this is that *changing options when a cursor is open can lead to unreliable results*. For example, changing DATE_FORMAT may not change the format for the next row when a cursor is opened. Depending on the way the cursor is being retrieved, it may take several rows before the change works its way to the user.

General database options

This section lists all database options.

OPTION	VALUES	DEFAULT
"AUDITING option" on page 140	ON, OFF	OFF
"BACKGROUND_PRIORITY option" on page 144	ON, OFF	OFF
"BLOCKING option" on page 145	ON, OFF	ON
"CHECKPOINT_TIME option" on page 147	number of minutes	60
"COOPERATIVE_COMMITS option" on page 149	ON, OFF	ON
"COOPERATIVE_COMMIT_TIMEOUT option" on page 150	integer	250
"DATE_FORMAT option" on page 150	string	'YYYY-MM-DD'
"DATE_ORDER option" on page 152	'YMD', 'DMY', 'MDY'	'YMD'
"DEFAULT_TIMESTAMP_INCREMENT option" on page 152	integer	1
"DELAYED_COMMITS option" on page 153	ON, OFF	OFF
"DELAYED_COMMIT_TIMEOUT option" on page 153	integer	500
"EXTENDED_JOIN_SYNTAX option" on page 155	ON/OFF	ON
"ISOLATION_LEVEL option" on page 159	0, 1, 2, 3	0
"JAVA_HEAP_SIZE option" on page 160	number of bytes	2000000
"JAVA_NAMESPACE_SIZE option" on page 160	number of bytes	4000000
"LOGIN_MODE option" on page 161	STANDARD, MIXED, INTEGRATED	STANDARD
"LOGIN_PROCEDURE option" on page 161	string	sp_login_environment
"MAX_CURSOR_COUNT option" on page 162	Integer	50
"MAX_STATEMENT_COUNT option" on page 162	Integer	50

OPTION	VALUES	DEFAULT
"MIN_PASSWORD_LENGTH option" on page 163	Integer	0
"PRECISION option" on page 168	number of digits	30
"PREFETCH option" on page 169	ON, OFF	ON
"RECOVERY_TIME option" on page 170	number of minutes	2
"ROW_COUNTS option" on page 170	ON, OFF	OFF
"SCALE option" on page 172	number of digits	6
"THREAD_COUNT option" on page 174	number of threads	0
"TIME_FORMAT option" on page 175	string	'HH:NN:ss.SSS'
"TIMESTAMP_FORMAT option" on page 175	string	'YYYY-MM-DD HH:NN:ss.SSS'
"WAIT_FOR_COMMIT option" on page 178	ON, OFF	OFF

Transact-SQLcompatibility options

The following options allow Adaptive Server Anywhere behavior to be made compatible with Adaptive Server Enterprise, or to support both old behavior and allow ISO SQL/92 behavior.

For further compatibility with Adaptive Server Enterprise, some of these options can be set for the duration of the current connection using the Transact-SQL SET statement instead of the Adaptive Server Anywhere SET OPTION statement. For a listing, see "SET statement" on page 548.

Default settings

The default setting for some of these options differs from the Adaptive Server Enterprise default setting. For compatibility, you should explicitly set the options to ensure compatible behavior.

When a connection is made using the Open Client or JDBC interfaces, some option settings are explicitly set for the current connection to be compatible with Adaptive Server Enterprise. These options are listed in the following table.

For information on how the settings are made, see "System and catalog stored procedures" on page 753.

Option	Setting
ALLOW_NULLS_BY_DEFAULT	OFF
ANSI_BLANKS	ON
ANSINULL	OFF
CHAINED	OFF
CONTINUE_AFTER_RAISERROR	ON
DATE_FORMAT	YYYY-MM-DD
DATE_ORDER	MDY
ESCAPE_CHARACTER	OFF
FLOAT_AS_DOUBLE	ON
ISOLATION_LEVEL	1
	CONDITIONAL
QUOTED_IDENTIFIER	OFF
TSQL_VARIABLES	ON
TSQL_HEX_CONSTANT	ON
TIME_FORMAT	HH:NN:SS.SSS
TIMESTAMP_FORMAT	YYYY-MM-DD HH:NN:SS.SSS

OPTION	VALUES	DEFAULT
"ALLOW_NULLS_BY_DEFAULT option" on page 140	ON, OFF	ON
"ANSI_BLANKS option" on page 140	ON, OFF	OFF
"ANSI_CLOSE_CURSORS_ON_ROLLBACK option" on page 141	ON, OFF	OFF
"ANSI_INTEGER_OVERFLOW option" on page 141	ON, OFF	ON
"ANSINULL option" on page 142	ON, OFF	ON
"ANSI_PERMISSIONS option" on page 142	ON, OFF	ON
"AUTOMATIC_TIMESTAMP option" on page 144	ON, OFF	OFF
"CHAINED option" on page 146	ON, OFF	ON
"CLOSE_ON_ENDTRANS option" on page 147	ON, OFF	ON
"CONTINUE_AFTER_RAISE_ERROR option" on page 148	ON,OFF	OFF
"CONVERSION_ERROR option" on page 149	ON,OFF	ON
"DIVIDE_BY_ZERO_ERROR option" on page 155	ON, OFF	OFF
"FIRE_TRIGGERS option" on page 156	ON, OFF	ON
"FLOAT_AS_DOUBLE option" on page 156	ON, OFF	OFF
"NEAREST_CENTURY option" on page 163	integer	50
"NON_KEYWORDS option" on page 164	comma-separated keywords list	No keywords turned off
"PERCENT_AS_COMMENT option" on page 167	ON, OFF	ON
"QUERY_PLAN_ON_OPEN option" on page 169	ON, OFF	OFF
"QUOTED_IDENTIFIER option" on page 170	ON, OFF	ON

OPTION	VALUES	DEFAULT
"RI_Trigger_time option" on page 172	BEFORE, AFTER	AFTER
"SQL_FLAGGER_ERROR_LEVEL option" on page 172	E, I, F, W	W
"SQL_FLAGGER_WARNING_LEVEL option" on page 173	E, I, F, W	W
"STRING_RTRUNCATION option" on page 173	ON, OFF	OFF
"TEXTSIZE option" on page 174	integer	32765
"TSQL_HEX_CONSTANT option" on page 177	ON, OFF	OFF
"TSQL_VARIABLES option" on page 177	ON, OFF	OFF

Replication options

The following options are included to provide control over replication behavior. Some replication options are useful for SQL Remote replication, and some are relevant for use with Sybase Replication Server.

OPTION	VALUES	DEFAULT
"BLOB_THRESHOLD option" on page 145	Integer	256
"DELETE_OLD_LOGS option" on page 154	ON, OFF	OFF
"REPLICATE_ALL option" on page 171	ON, OFF	OFF
"REPLICATION_ERROR option" on page 171	Procedure-name	(no procedure)
"SUBSCRIBE_BY_REMOTE option" on page 174	ON, OFF	ON
"VERIFY_ALL_COLUMNS option" on page 177	ON, OFF	OFF
"VERIFY_THRESHOLD option" on page 178	integer	1000

Interactive SQL options

Syntax 1

SET [**TEMPORARY**] **OPTION**
... [*userid*. | **PUBLIC**.]*option-name* = [*option-value*]

Syntax 2

SET PERMANENT

Syntax 3

SET

Parameters

userid: *identifier*, *string* or *host-variable*

option-name: *identifier*, *string* or *host-variable*

option-value: *host-variable* (indicator allowed), *string*, *identifier*, or *number*

Description

SET PERMANENT (syntax 2) stores all current Interactive SQL options in the SYSOPTIONS system table. These settings are automatically established every time Interactive SQL is started for the current user ID.

Syntax 3 is used to display all of the current option settings. If any options have temporary settings for Interactive SQL or the database server, these are displayed; otherwise, the permanent option settings are displayed.

OPTION	VALUES	DEFAULT
"AUTO_COMMIT option" on page 143	ON, OFF	OFF
"AUTO_REFETCH option" on page 143	ON, OFF	ON
"BELL option" on page 145	ON, OFF	ON
"CHAR_OEM_TRANSLATION option" on page 146	ON, OFF, DETECT	DETECT
"COMMAND_DELIMITER option" on page 147	string	';'
"COMMIT_ON_EXIT option" on page 148	ON, OFF	ON
"DESCRIBE_JAVA_FORMAT option" on page 154	Varchar, binary	Varchar
"ECHO option" on page 155	ON, OFF	ON
"HEADINGS option" on page 157	ON, OFF	ON
"INPUT_FORMAT option" on page 157	ASCII DBASE DBASEII DBASEIII DIF FIXED FOXPRO LOTUS WATFILE	ASCII
"ISQL_LOG option" on page 159	file-name	(empty string)

OPTION	VALUES	DEFAULT
"NULLS option" on page 164	ON, OFF	'(NULL)'
"ON_ERROR option" on page 165	STOP CONTINUE PROMPT EXIT NOTIFY_CONTINUE NOTIFY_STOP NOTIFY_EXIT	PROMPT
"OUTPUT_FORMAT option" on page 166	TEXT ASCII FIXED DIF DBASEII DBASEIII FOXPRO LOTUS SQL WATFILE	ASCII
"OUTPUT_LENGTH option" on page 167	integer	0
"STATISTICS option" on page 173	0,3,4,5,6	3
"TRUNCATION_LENGTH option" on page 176	integer	30

Alphabetical list of options

This section lists options alphabetically

AUDITING option

Function Enables and disables auditing in the database.

Allowed values ON, OFF

Scope Can be set for the PUBLIC group only. Takes effect immediately. DBA permissions are required to set this option.

Default OFF

Description This option switches auditing on and off.

Auditing is the recording of detailed information about many events in the database in the transaction log. Auditing provides some security features, at the cost of some performance.

Example

- Turn on auditing

```
SET OPTION PUBLIC.AUDITING = 'ON'
```

ALLOW_NULLS_BY_DEFAULT option

Function Controls whether new columns that are created without specifying either NULL or NOT NULL are allowed contain NULL values.

Allowed values ON, OFF

Default ON

OFF for Open Client and JDBC connections

Description The ALLOW_NULLS_BY_DEFAULT option is included for Transact-SQL compatibility.

For more information, see "Setting options for Transact-SQL compatibility" on page 794 of the book *Adaptive Server Anywhere User's Guide*.

ANSI_BLANKS option

Function Controls behavior when character data is truncated at the client side.

Allowed values ON, OFF

Default OFF

ON for Open Client and JDBC connections

Description The ANSI_BLANKS option has no effect unless the database was created with the `-b` command-line option. It forces a truncation error whenever a value of data type CHAR(*N*) is read into a C char(*M*) variable for values of *N* greater than or equal to *M*. With ANSI_BLANKS set to OFF, a truncation error occurs only when at least one non-blank character is truncated.

If ANSI_BLANKS is ON, when you supply a value of data type DT_STRING, the **sqllen** field must be set to the length of the value, including space for the terminating null character. With ANSI_BLANKS off, the length is determined solely by the position of the null character.

The ANSI_BLANKS setting persists for the life of a connection. Changing it after a connection has been established does not affect that connection.

ANSI_CLOSE_CURSORS_ON_ROLLBACK option

Function Controls whether cursors that were opened WITH HOLD are closed when a ROLLBACK is performed.

Allowed values ON, OFF

Default OFF

Description The draft SQL/3 standard requires all cursors be closed when a transaction is rolled back. By default, on a rollback Adaptive Server Anywhere closes only those cursors that were opened without a WITH HOLD clause. This option allows you to force closure of all cursors.

The CLOSE_ON_ENDTRANS option overrides the ANSI_CLOSE_CURSORS_ON_ROLLBACK option.

ANSI_INTEGER_OVERFLOW option

Function Controls what happens when an arithmetic expression causes an integer overflow error.

Allowed values ON, OFF

Default OFF

Description The ISO SQL/92 standard requires integer overflow should result in an SQLSTATE = 22003 - overflow error. Adaptive Server Anywhere behavior was previously different from this. The ANSI_INTEGER_OVERFLOW option can be set to OFF to maintain compatibility with previous releases of the software.

ANSINULL option

Function Controls the interpretation of = and != with NULL values.

Allowed values ON, OFF

Default ON

Description With ANSINULL ON, any comparisons with NULL using = or != are unknown.

Also, aggregate functions on columns that icontain NULL values cause the warning 'null value eliminated in aggregate function' (SQLSTATE=01003).

Setting ANSINULL to OFF allows comparisons with NULL to yield results that are not unknown, for compatibility with Adaptive Server Enterprise, and turns off the warning.

ANSI_PERMISSIONS option

Function Controls permissions checking for DELETE and UPDATE statements.

Allowed values ON, OFF

Default ON

Description With ANSI_PERMISSIONS ON, the SQL/92 permissions requirements for DELETE and UPDATE statements are checked. The default value is OFF in Adaptive Server Enterprise. The following table outlines the differences.

SQL Statement	Permissions Required with ansi_permissions off	Permissions Required with ansi_permissions on
UPDATE	UPDATE permission on the columns where values are being set	UPDATE permission on the columns where values are being set SELECT permission on all columns appearing in the WHERE clause. SELECT permission on all columns on the right side of the set clause.
DELETE	DELETE permission on the table	DELETE permission on the table. SELECT permission on all columns appearing in the WHERE clause

The ANSI_PERMISSIONS option can be set only for the PUBLIC group. No private settings are allowed.

AUTO_COMMIT option [ISQL

Function Controls whether a COMMIT is performed after each statement.

Allowed values ON, OFF

Default OFF

Description If AUTO_COMMIT is *on*, a database COMMIT is performed after each successful statement. If the COMMIT fails, you have the option to execute additional SQL statements and perform the COMMIT again, or execute a ROLLBACK statement.

By default, a COMMIT or ROLLBACK is performed only when the user issues a COMMIT or ROLLBACK statement or a SQL statement that causes an automatic commit (such as the CREATE TABLE statement).

AUTO_REFETCH option [ISQL

Function Controls whether query results are fetched again after deletes, updates, and inserts.

Allowed values ON, OFF

Default ON

Description If AUTO_REFETCH is *on*, the current query results that are displayed in the Data window will be refetched from the database after **any** INSERT, UPDATE or DELETE statement. Depending on how complicated the query is, this may take some time. For this reason, it can be turned *off*.

AUTOMATIC_TIMESTAMP option

Function Controls interpretation of new columns with the TIMESTAMP data type.

Allowed values ON, OFF

Default OFF

Description Controls whether any new columns with the TIMESTAMP data type that do not have an explicit default value defined are given a default value of the Transact-SQL **timestamp** value as a default. The AUTOMATIC_TIMESTAMP option is included for Transact-SQL compatibility. The default is OFF.

For more information, see "Setting options for Transact-SQL compatibility" on page 794 of the book *Adaptive Server Anywhere User's Guide*.

BACKGROUND_PRIORITY option

Function To limit impact on the performance of connections other than the current connection.

Allowed Values ON or OFF

Scope Can be set for an individual connection or the PUBLIC group. Takes effect immediately.

If you set this option temporarily, that setting applies to the current connection only. Different connections under the same user ID can have different settings for this option.

Default OFF

Description When set to ON, it requests that the current connection have minimal impact on the performance of other connections. This option allows tasks for which responsiveness is critical to coexist with other tasks for which performance is not as important.

BELL option [ISQL

Function Controls whether the bell sounds when an error occurs.

Allowed values ON, OFF

Default ON

Description Set this option according to your preference.

BLOB_THRESHOLD option

Function Controls the size of value that the Message Agent treats as a long object (blob).

Allowed values Integer, in kilobytes

Default 256

Description Any value longer than the BLOB_THRESHOLD option is replicated as a blob. That is, it is broken into pieces and replicated in chunks, before being reconstituted by using a SQL variable and concatenating the pieces at the recipient site.

If you set BLOB_THRESHOLD to a high value in remote Adaptive Server Anywhere databases, blobs are not broken into pieces, and operations can be applied to Adaptive Server Enterprise by the Message Agent. Each SQL statement must fit within a message, so this only allows replication of small blobs.

BLOCKING option

Function Controls the behavior in response to locking conflicts.

Allowed Values ON or OFF

Scope Can be set for an individual connection or the PUBLIC group. Takes effect immediately.

Default ON

Description If BLOCKING is ON, any transaction that attempts to write must wait until every conflicting transaction releases its read lock. At that time, the write goes through. If BLOCKING is OFF, the transaction that attempts to write receives an error.

For more information, see "Transaction blocking and deadlock" on page 416 of the book *Adaptive Server Anywhere User's Guide*.

CHAINED option

Function Controls transaction mode in the absence of a BEGIN TRANSACTION statement.

Allowed values ON, OFF

Default ON

OFF for Open Client and JDBC connections

Description Controls the Transact-SQL transaction mode. In Unchained mode (CHAINED = OFF), each statement is committed individually unless an explicit BEGIN TRANSACTION statement is executed to start a transaction. In chained mode (CHAINED = ON) a transaction is implicitly started before any data retrieval or modification statement.

CHAR_OEM_TRANSLATION option [ISQL

Function Controls whether ANSI-to-OEM code page translation is carried out.

Allowed values ON, OFF, DETECT

Default DETECT

Description Each time Interactive SQL connects to a database or uses SET CONNECTION, it determines the setting of Char_OEM_Translation.

If the option is set to ON or OFF, translation is set accordingly. If the option is set to DETECT (the recommended and default setting), Interactive SQL fetches the collation label from the database and examines it. If the collation label starts with a string that generally indicates an ANSI code page, then Interactive SQL turns off translation.

The strings that indicate ANSI code pages are as follows:

- WIN_
- ISO_
- SJIS
- EUC_
- UTF
- EBCDIC
- 813
- 819

For all other prefixes, translation is turned on. When the option is set to DETECT, Interactive SQL displays a message in the status window indicating the collation label and the display translation setting.

CHECKPOINT_TIME option

Function Set the maximum number of minutes that the database server will run without doing a checkpoint.

Allowed Values Integer

Scope Can be set only for the PUBLIC group. You must shut down and restart the database server for the change to take effect.

Default 60

Description This option is used with the "RECOVERY_TIME option" on page 170 to decide when checkpoints should be done.

For information on checkpoints, see "The checkpoint log" on page 555 of the book *Adaptive Server Anywhere User's Guide*.

CLOSE_ON_ENDTRANS option

Function Controls the closing of cursors at the end of a transaction.

Allowed values ON, OFF

Default ON

Description When CLOSE_ON_ENDTRANS is set to ON, cursors are closed whenever a transaction is committed unless the cursor was opened WITH HOLD. The behavior when a transaction is rolled back is governed by the ANSI_CLOSE_CURSORS_AT_ROLLBACK option.

When CLOSE_ON_ENDTRANS is set to OFF, cursors are not closed at either a commit or a rollback, regardless of the ANSI_CLOSE_CURSORS_AT_ROLLBACK option setting or whether the cursor was opened WITH HOLD or not.

Setting this to OFF provides Adaptive Server Enterprise compatible behavior.

COMMAND_DELIMITER option [ISQL

Function Sets the string that indicates the end of a statement in Interactive SQL.

Allowed values String

Default Semi-colon (;)

Description If the command delimiter is set to a string beginning with a character that is valid in identifiers, the command delimiter must be preceded by a space.

COMMIT_ON_EXIT option [ISQL

Function Controls behavior when Interactive SQL disconnects or terminates.

Allowed values ON, OFF

Default ON

Description Controls whether a COMMIT or ROLLBACK is done when you leave Interactive SQL. When COMMIT_ON_EXIT is set to ON, a COMMIT is done; otherwise a ROLLBACK is done.

CONTINUE_AFTER_RAISE_ERROR option

Function Controls behavior following a RAISERROR statement.

Allowed values ON, OFF

Default OFF

ON for Open Client and JDBC connections

Description

Deprecated option
This option is deprecated in favor of the ON_TSQL_ERROR option.

For more information, see "ON_TSQL_ERROR option" on page 165.

The RAISERROR statement is used within procedures and triggers to generate an error. When this option is set to OFF, the execution of the procedure or trigger is stopped whenever the RAISERROR statement is encountered.

If you set the CONTINUE_AFTER_RAISERROR switch to ON, the RAISERROR statement no longer signals an execution-ending error. Instead, the RAISERROR status code and message are stored and the most recent RAISERROR is returned when the procedure completes. If the procedure which caused the RAISERROR was called from another procedure, the RAISERROR is not returned until the outermost calling procedure terminates.

Intermediate RAISERROR statuses and codes are lost after the procedure terminates. If, at return time, an error occurs along with the RAISERROR, then the information for the new error is returned and the RAISERROR information is lost. The application can query intermediate RAISERROR statuses by examining the @@error global variable at different execution points.

CONVERSION_ERROR option

Function Controls the reporting of data type conversion failures on fetching information from the database.

Allowed values ON, OFF

Default ON

Description This option controls whether data type conversion failures, when data is fetched from the database or inserted into the database, are reported by the database as errors (CONVERSION_ERROR set to ON) or as a warning (CONVERSION_ERROR set to OFF).

When CONVERSION_ERROR is set to ON, the SQLE_CONVERSION_ERROR error is generated. If the option is set to OFF, the warning SQLE_CANNOT_CONVERT is produced.

If conversion errors are reported as warnings only, the NULL value is used in place of the value that could not be converted. In Embedded SQL, an indicator variable is set to -2 for the column or columns that cause the error.

COOPERATIVE_COMMITS option

Function Controls when commits are written to disk.

Allowed Values ON or OFF

Scope Can be set for an individual connection or the PUBLIC group. Takes effect immediately.

Default ON

Description If COOPERATIVE_COMMITS is set to OFF, a COMMIT is written to disk as soon as the database server receives it, and the application is then allowed to continue.

If COOPERATIVE_COMMITS is set to ON (the default), the database server does not immediately write the COMMIT to the disk. Instead, the application waits for up to the maximum length set by the COOPERATIVE_COMMIT_TIMEOUT option for something else to put on the pages before they are written to disk.

Setting COOPERATIVE_COMMITS to ON, and increasing the COOPERATIVE_COMMIT_TIMEOUT setting, increases overall database server throughput by cutting down the number of disk I/Os, but at the expense of a longer turnaround time for each individual connection.

If both COOPERATIVE_COMMITS and DELAYED_COMMITS are set to ON, and the COOPERATIVE_COMMIT_TIMEOUT interval passes without the pages getting written, the application is resumed (as if the commit had worked), and the remaining interval (DELAYED_COMMIT_TIMEOUT - COOPERATIVE_COMMIT_TIMEOUT) is used as a DELAYED_COMMIT interval. The pages will then be written, even if they are not full.

COOPERATIVE_COMMIT_TIMEOUT option

Function Governs when a COMMIT entry in the transaction log is written to disk.

Allowed Values Integer, in milliseconds

Scope Can be set for an individual connection or the PUBLIC group. Takes effect immediately.

Default 250

Description This option has meaning only when COOPERATIVE_COMMITS is set to ON. The database server waits for the specified number of milliseconds for other connections to fill a page of the log before writing to disk. The default setting is 250 milliseconds.

DATE_FORMAT option

Function Sets the format for dates retrieved from the database.

Allowed Values

String

Scope

Can be set for an individual connection or the PUBLIC group. Takes effect immediately.

Default

'YYYY-MM-DD'. This corresponds to ISO date format specifications.

Description

The format is a string using the following symbols:

Symbol	Description
yy	Two digit year
yyyy	Four digit year
mm	Two digit month, or two digit minutes if following a colon(as in hh:mm)
mmm[m...]	Character short form for months—as many characters as there are m's
d	Single digit day of week, (0 = Sunday, 6 = Saturday)
dd	Two digit day of month
ddd[d...]	Character short form for day of the week
hh	Two digit hours
nn	Two digit minutes
ss[.ss..]	Seconds and parts of a second
aa	AM or PM (12 hour clock)
pp	PM if needed (12 hour clock)
jjj	Day of the year, from 1 to 366.

Each symbol is substituted with the appropriate data for the date that's being formatted. Any format symbol that represents character rather than digit output can be put in upper case which causes the substituted characters to also be in upper case. For numbers, using mixed case in the format string suppresses leading zeros.

You can control the padding of numbers by changing the case of the symbols. Same-case symbols (MM, mm, DD, dd) all pad number with zeroes. Mixed case (Mm, mM, Dd, or dD) cause the number to not be zero padded: the value takes as much room as required. For example

```
SELECT dateformat( '1998/01/01', 'yyyy/Mm/Dd')
```

returns the following value:

```
1998/1/1
```

Examples

- The following table illustrates DATE_FORMAT settings, together with the output from the following statement, executed on Thursday May 21, 1998

```
SELECT CURRENT DATE
```

DATE_FORMAT	SELECT CURRENT DATE
yyyy/mm/dd/ddd	1998/05/21/thu
jjj	141
mmm yyyy	may 1998
mm-yyyy	05-1998

DATE_ORDER option

Function Controls the interpretation of date formats.

Allowed Values 'MDY', 'YMD', or 'DMY'

Scope Can be set for an individual connection or the PUBLIC group. Takes effect immediately.

Default YMD. This corresponds to ISO date format specifications.

For Open Client and JDBC connections, the default is set to MDY.

Description The database option DATE_ORDER is used to determine whether 10/11/12 is Oct 11 1912, Nov 12 1910, or Nov 10 1912. The option can have the value 'MDY', 'YMD', or 'DMY'.

DEFAULT_TIMESTAMP_INCREMENT option

Function Specifies the number of microseconds to add to a column of type TIMESTAMP in order to keep values in the column unique.

Allowed Values Integer

Scope Can be set for an individual connection or the PUBLIC group. Takes effect immediately.

Default 1

Description Since a TIMESTAMP value is precise to six decimal places in Adaptive Server Anywhere, by default 1 microsecond (0.000001 of a second) is added to differentiate between two identical TIMESTAMP values.

Some software, such as Microsoft Access, truncates TIMESTAMP values to three decimal places, making valid comparisons a problem. Setting this value to a larger number, such as 0.001 of a second (an option value of 100), alleviates this problem.

DELAYED_COMMITS option

Function Determines when the server returns control to an application following a COMMIT.

Allowed Values ON or OFF

Scope Can be set for an individual connection or the PUBLIC group. Takes effect immediately.

Default OFF. This corresponds to ISO COMMIT behavior.

Description When set to ON, the database server replies to a COMMIT statement immediately instead of waiting until the transaction log entry for the COMMIT has been written to disk. When set to OFF, the application must wait until the COMMIT is written to disk.

When this option is ON, the log is written to disk when the log page is full or according to the DELAYED_COMMIT_TIMEOUT option setting, whichever is first. There is a slight chance that a transaction may be lost even though committed if a system failure occurs after the server replies to a COMMIT, but before the page is written to disk. Setting DELAYED_COMMITS to ON, and the DELAYED_COMMIT_TIMEOUT option to a high value, promotes a quick response time at the cost of security.

If both COOPERATIVE_COMMITS and DELAYED_COMMITS are set to ON, and if the COOPERATIVE_COMMIT_TIMEOUT interval passes without the pages getting written, the application is resumed (as if the commit had worked), and the remaining interval (DELAYED_COMMIT_TIMEOUT - COOPERATIVE_COMMIT_TIMEOUT) is used as a DELAYED_COMMIT interval, after which the pages will be written, even if they are not full.

DELAYED_COMMIT_TIMEOUT option

Function Determines when the server returns control to an application following a COMMIT.

Allowed Values Integer, in milliseconds.

Scope Can be set for an individual connection or the PUBLIC group. Takes effect immediately.

Default 500

Description This option has meaning only when DELAYED_COMMITS is set to ON. it governs when a COMMIT entry in the transaction log is written to disk. With DELAYED_COMMITS set to ON, the database engine waits for the number of milliseconds set in the DELAYED_COMMIT_TIMEOUT option for other connections to fill a page of the log before writing the current page contents to disk.

For more information, see "DELAYED_COMMITS option" on page 153.

DELETE_OLD_LOGS option

Function Controls whether transaction logs are deleted when their messages have been replicated.

Allowed values ON, OFF

Default OFF

Description This option is used by SQL Remote and by the Adaptive Server Anywhere Replication Agent. The default setting is OFF. When it is set to ON, the Message Agent (DBREMOTE) deletes each old transaction log when all the changes it contains have been sent and confirmed as received.

DESCRIBE_JAVA_FORMAT option [ISQL]

Function Controls whether Java objects are interpreted as strings (for display) or as binary (for loading and unloading).

Allowed values **varchar**, **binary**

Default OFF

Description When set to **varchar**, Interactive SQL converts all data fetched from the database to strings. The database server calls the `toString()` method on Java columns to provide formatted output for the Interactive SQL data window.

When set to **binary**, Interactive SQL does no data conversion.

DIVIDE_BY_ZERO_ERROR option

Function Controls the reporting of division by zero.

Allowed values ON, OFF

Default ON

Description This option indicates whether division by zero is reported as an error. If the option is set ON, then division by zero results in an error with SQLSTATE 22012.

If the option is set OFF, division by zero is not an error. Instead, a NULL is returned.

ECHO option [ISQL

Function Controls whether statements are echoed before they are executed.

Allowed values ON, OFF

Default ON

Description This option is most useful when you use the READ statement to execute a Interactive SQL command file.

ESCAPE_CHARACTER option

This option is reserved for system use. Do not change the setting of this option.

EXTENDED_JOIN_SYNTAX option

Function Controls whether queries with an ambiguous syntax for multi-table joins are allowed, or reported as an error.

Allowed values ON, OFF

Default ON

Description This option reports a syntax error for those queries containing outer joins that have ambiguous syntax due to the presence of duplicate correlation names on a null-supplying table.

The following join clause illustrates the kind of query that is reported.

```
( R left outer join T , T  join S on ( C1 ) )
```

where C1 is a condition. If the option is set to ON, this query is interpreted as follows.

```
( R left outer join T on ( C1 ) ) join S on ( C2 )
```

where C1 and C2 are conditions.

FIRE_TRIGGERS option

Function Controls whether triggers are fired in the database.

Allowed values ON, OFF

Default ON

Description When set to ON, triggers are fired. When set to OFF, no triggers are fired, including referential integrity triggers (such as cascading updates and deletes). Only a user with DBA authority can set this option. The option is overridden by the `-gf` command-line option, which turns off all trigger firing regardless of the FIRE_TRIGGERS setting.

This option is relevant when replicating data from Adaptive Server Enterprise to Adaptive Server Anywhere, because all actions from Adaptive Server Enterprise transaction logs are replicated to Adaptive Server Anywhere, including actions carried out by triggers.

FLOAT_AS_DOUBLE option

Function Controls the interpretation of the FLOAT keyword.

Allowed values ON, OFF

Default OFF

ON for Open Client and JDBC connections

Description The FLOAT_AS_DOUBLE option makes the FLOAT keyword behave like Adaptive Server Enterprise's FLOAT keyword when a precision is not specified.

When enabled (set to ON), all occurrences of the keyword FLOAT are interpreted as equivalent to the keyword DOUBLE within SQL statements.

By default, Adaptive Server Anywhere FLOAT values are interpreted by Adaptive Server Enterprise as REAL values. Since Adaptive Server Enterprise treats its own FLOAT values as DOUBLE, enabling this option makes Adaptive Server Anywhere treat FLOAT values in the same way Enterprise treats FLOAT values.

REAL values are four bytes; DOUBLE values are eight bytes. According to the ANSI SQL/92 specification, FLOAT can be interpreted based on the platform. It is up to the database to decide what size the value is, so long as it can handle the necessary precision. Adaptive Server Enterprise and Adaptive Server Anywhere exhibit different default behavior.

The FLOAT_AS_DOUBLE option takes effect only when no precision is specified. For example, the following statement is not affected by the option setting:

```
create table t1(
   c1 float(5)
)
```

The following statement is affected by the option setting:

```
create table t2(
   c1 float)
// affected by option setting
```

HEADINGS option [ISQL

Function Controls whether headings will be displayed for the results of a SELECT statement.

Allowed values ON, OFF

Default ON

Description Set this option according to your preference.

INPUT_FORMAT option [ISQL

Function Sets the default data format expected by the INPUT statement.

Allowed values String. See below.

Default ASCII

Description

Certain file formats contain information about column names and types. Using this information, the INPUT statement will create the database table if it does not already exist. This is a very easy way to load data into the database. The formats that have enough information to create the table are: DBASEII, DBASEIII, DIF, FOXPRO, LOTUS, and WATFILE.

Allowable input formats are:

- **ASCII** Input lines are assumed to be ASCII characters, one row per line, with values separated by commas. Alphabetic strings may be enclosed in apostrophes (single quotes) or quotation marks (double quotes). Strings containing commas must be enclosed in either single or double quotes. If single or double quotes are used, double the quote character to use it within the string. Optionally, you can use the DELIMITED BY clause to specify a different delimiter string than the default, which is a comma(,).

  Three other special sequences are also recognized. The two characters \n represent a newline character, \\ represents a single backslash character, and the sequence \xDD represents the character with hexadecimal code DD.

- **DBASE** The file is in dBASE II or dBASE III format. Interactive SQL will attempt to determine which format, is based on information in the file. If the table doesn't exist, it will be created.

- **DBASEII** The file is in dBASE II format. If the table doesn't exist, it will be created.

- **DBASEIII** The file is in dBASE III format. If the table doesn't exist, it will be created.

- **DIF** The file is in Data Interchange Format. If the table doesn't exist, it will be created.

- **FIXED** Input lines are in fixed format. The width of the columns can be specified using the COLUMN WIDTHS clause. If they are not specified, column widths in the file must be the same as the maximum number of characters required by any value of the corresponding database column's type.

- **FOXPRO** The file is in FoxPro format. The FoxPro memo field is different than the dBASE memo field. If the table doesn't exist, it will be created.

- **LOTUS** The file is a Lotus WKS format worksheet. INPUT assumes that the first row in the Lotus WKS format worksheet consists of column names. If the table doesn't exist, it will be created. In this case, the types and sizes of the columns created may not be correct because the information in the file pertains to a cell, not to a column.

- **WATFILE** The input will be a WATFILE file. If the table doesn't exist, it will be created.

ISQL_LOG option [ISQL

Function Controls logging behavior.

Allowed values String containing a file name.

Default Empty string.

Description If ISQL_LOG is set to a non-empty string, all Interactive SQL statements are added to the end of the named file. Otherwise, if ISQL_LOG is set to the empty string, Interactive SQL statements are not logged.

> **Individual session only**
> This option logs an individual Interactive SQL session only. See "Backup and Data Recovery" on page 553 of the book *Adaptive Server Anywhere User's Guide* for a description of the transaction log that logs all changes to the database by all users.

ISOLATION_LEVEL option

Function Controls the locking isolation level.

Allowed Values 0, 1, 2, or 3

Scope Can be set for an individual connection or the PUBLIC group. Takes effect immediately.

Default 0

1 for Open Client and JDBC connections

Description This option controls the locking isolation level as follows.

- **0** Allow dirty reads, nonrepeatable reads, and phantom rows.
- **1** Prevent dirty reads. Allow nonrepeatable reads and phantom rows.
- **2** Prevent dirty reads and guarantee repeatable reads. Allow phantom rows.
- **3** Serializable. Do not allow dirty reads, guarantee repeatable reads, and do not allow phantom rows.

☞ For more information, see "Isolation levels and consistency" on page 386 of the book *Adaptive Server Anywhere User's Guide*.

JAVA_HEAP_SIZE option

Function To limit the memory used by Java applications for a connection.

Allowed values Integer

Scope Can be set for an individual connection or the PUBLIC group. Takes effect immediately. DBA permissions are required to set this option for *any* connection.

Default 1000000

Description This option sets the maximum size (in bytes) of the memory that is allocated to Java applications on a per-connection basis. Per-connection memory allocations typically consist of the user's working set of allocated Java variables and Java application stack space.

While a Java application is executing on a connection, the per-connection allocations come out of the fixed cache of the database server, so it is important that a run-away Java application is prevented from using up too much memory.

JAVA_NAMESPACE_SIZE option

Function To limit the memory used by Java applications for a connection.

Allowed values Integer

Scope Can be set only for the PUBLIC group. Takes effect immediately.

Default 4000000

Description This option sets the maximum size (in bytes) of the memory that is allocated to Java applications on a per-database basis.

Per-database memory allocations include Java class definitions. Because class definitions are effectively read-only, they are shared between connections. Consequently, their allocations come right out of the fixed cache, and this option sets a limit on the size of these allocations.

LOGIN_MODE option

Function Controls the use of integrated logins for the database.

Allowed Values Standard, Mixed, or Integrated

Scope Can be set only for the PUBLIC group. Takes effect immediately.

Default Standard

Description This option specifies whether integrated logins are permitted. The following values are accepted (the values are case insensitive):

- **Standard** This is the default setting, which does not permit integrated logins. An error occurs if an integrated login connection is attempted.
- **Mixed** With this setting, both integrated logins and standard logins are allowed.
- **Integrated** With this setting, all logins to the database must be made using integrated logins.

> ***Caution***
> *Setting the LOGIN_MODE database option to Integrated restricts connections to only those users who have been granted an integrated login mapping. Attempting to connect with a user ID and password generates an error. The only exceptions to this are users with DBA authority (full administrative rights).*

For more information on integrated logins see "Using integrated logins" on page 58 of the book *Adaptive Server Anywhere User's Guide*

LOGIN_PROCEDURE option

Function A login procedure that sets connection compatibility options at startup. By default the procedure calls the **sp_login_environment** procedure to determine which options to set.

Allowed Values String

Scope Can be set for an individual connection or the PUBLIC group. Takes effect immediately.

Default sp_login_environment

Description This login procedure calls the sp_login_environment procedure at run time to determine the database connection settings.

You can customize the default database option settings by creating a new procedure and setting LOGIN_PROCEDURE to call the new procedure. You should not edit either sp_login_procedure or sp_tsql_environment.

MAX_CURSOR_COUNT option

Function

A resource governor to limit the maximum number of cursors that a connection can use at once.

Allowed Values

Integer

Scope

Can be set for an individual connection or the PUBLIC group. Takes effect immediately. DBA permissions are required to set this option for *any* connection.

Default

50

Description

This resource governor allows a DBA to limit the number of cursors per connection that a user can use. If an operation would exceed the limit for a connection, an error is generated, indicating that the governor for the resource has been exceeded.

If a connection executes a stored procedure, that procedure is executed under the permissions of the procedure owner. However, the resources used by the procedure are assigned to the current connection.

You can remove resource limits by setting the option to 0 (zero).

MAX_STATEMENT_COUNT option

Function

A resource governor to limit the maximum number of prepared statements that a connection can use at once.

Allowed Values

Integer

Scope

Can be set for an individual connection or the PUBLIC group. Takes effect immediately. DBA permissions are required to set this option for *any* connection.

Default

50

Description

This resource governor allows a DBA to limit the number of prepared statements per connection a user can use. If an operation would exceed the limit for a connection, an error is generated, indicating that the governor for the resource has been exceeded.

If a connection executes a stored procedure, that procedure is executed under the permissions of the procedure owner. However, the resources used by the procedure are assigned to the current connection.

You can remove resource limits by setting the option to 0 (zero).

MIN_PASSWORD_LENGTH option

Function

Sets the minimum length for new passwords in the database.

Allowed values

Integer, greater than or equal to zero.

The value is in bytes. For single-byte character sets, this is the same as the number of characters.

Scope

Can be set for the PUBLIC group. Takes effect immediately. DBA permissions are required to set this option.

Default

0 characters

Description

This option allows the database administrator to impose a minimum length on all new passwords for greater security. Existing passwords are not affected.

Example

- Set the minimum length for new passwords to 6 bytes.

```
SET OPTION PUBLIC.MIN_PASSWORD_LENGTH = 6
```

NEAREST_CENTURY option

Function

Controls the interpretation of two-digit years, in string-to-date conversions.

Allowed values

Integer between 0 and 100

Default

50 for databases created with Version 6 or later.

0 for databases created with Version 5.5 or earlier.

Description

This option controls the handling of two-digit years, when converting from strings to dates or timestamps.

The NEAREST_CENTURY setting is a numeric value that acts as a rollover point. Two digit years less than the value are converted to 20yy, while years greater than or equal to the value are converted to 19yy.

The historical Adaptive Server Anywhere behavior is to add 1900 to the year. Adaptive Server Enterprise behavior is to use the nearest century, so any year value *yy* is less than 50, the year is set to 20yy.

NON_KEYWORDS option

Function Turns off individual keywords, allowing their use as identifiers.

Allowed values String

Default The empty string.

Description This option turns off individual keywords, or all keywords introduced since a specific release of the product. This provides a way of ensuring that applications created with older versions of the product are not broken by new keywords. If you have an identifier in your database that is now a keyword, you can either add double quotes around the identifier in all applications or scripts, or turn off the keyword using the NON_KEYWORDS option.

In addition to specifying individual keywords, you can turn off all keywords since a specified release, using one of the following special values in the list of keywords:

```
keywords_4_0_d, keywords_4_0_c, keywords_4_0_b,
keywords_4_0_a, keywords_4_0, keywords_5_0_01,
keywords_5_0
```

The following statement prevents TRUNCATE and SYNCHRONIZE from being recognized as keywords:

```
SET OPTION NON_KEYWORDS = 'TRUNCATE, SYNCHRONIZE'
```

The following statement prevents all keywords introduced since release 4.0d from being recognized as keywords:

```
SET OPTION NON_KEYWORDS = 'keywords_4_0_d'
```

Each new setting of this option replaces the previous setting. The following statement clears all previous settings.

```
SET OPTION NON_KEYWORDS =
```

A side-effect of this options is that SQL statements that use a turned off keyword cannot be used: they produce a syntax error.

NULLS option [ISQL

Function Specifies how NULL values in the database are displayed.

Allowed values ON, OFF

Default (NULL)

Description Set this according to your preference.

ON_ERROR option [ISQL

Function Controls what happens if an error is encountered while reading statements from a command file.

Allowed values String. See below for allowed values.

Default PROMPT

Description Controls what happens if an error is encountered while reading statements from a command file, as follows:

- **STOP** Interactive SQL stops reading statements from the file and returns to the statement window for input.
- **PROMPT** Interactive SQL prompts the user to see if the user wishes to continue.
- **CONTINUE** The error is ignored and Interactive SQL continues reading statements from the command file. The INPUT statement continues with the next row, skipping the row that caused the error.
- **EXIT** Interactive SQL terminates.
- **NOTIFY_CONTINUE** The error is displayed in a message box with a single button. Execution continues once the button is clicked.
- **NOTIFY_STOP** The error is displayed in a message box with a single button. Execution of the script stops once the button is clicked.
- **NOTIFY_EXIT** The error is displayed in a message box with a single button. Interactive SQL terminates once the button is clicked.

ON_TSQL_ERROR option

Function Controls what happens if an error is encountered in a stored procedure/

Allowed values String. See below for allowed values.

Default CONDITIONAL

See also "CREATE PROCEDURE statement" on page 403
"CREATE PROCEDURE statement" on page 409
"Transact-SQL procedure language overview" on page 805 of the book *Adaptive Server Anywhere User's Guide*

Description Controls what happens if an error is encountered while executing a Transact-SQL stored procedure.

- **STOP** Stop execution immediately upon finding an error.

- **CONDITIONAL** If the procedure uses ON EXCEPTION RESUME, and the statement following the error handles the error, continue, otherwise exit.
- **CONTINUE** Continue execution, regardless of the following statement. If there are multiple errors, the first error encountered in the stored procedure is returned. This option most closely mirrors Adaptive Server Enterprise behavior.

This option deprecates CONTINUE_AFTER_RAISEERROR as a way of simulating Adaptive Server Enterprise behavior for stored procedures..

OUTPUT_FORMAT option [ISQL

Function Sets the output format for the data retrieved by the SELECT statement and redirected into a file, or output using the OUTPUT statement.

Allowed values String. See below for allowed values.

Default ASCII

Description The valid output formats are:

- **ASCII** The output is an ASCII format file with one row per line in the file. All values are separated by commas, and strings are enclosed in apostrophes (single quotes). The delimiter and quote strings can be changed using the DELIMITED BY and QUOTE clauses. If ALL is specified in the QUOTE clause, then all values (not just strings) will be quoted.

  Three other special sequences are also used. The two characters \n represent a newline character, \\ represents a single backslash character, and the sequence \xDD represents the character with hexadecimal code DD.
- **DBASEII** The output is a dBASE II format file with the column definitions at the top of the file. Note that a maximum of 32 columns can be output. Also, note that columns longer than 255 characters will be truncated in the file.
- **DBASEIII** The output is a dBASE III format file with the column definitions at the top of the file. Note that a maximum of 128 columns can be output. Also, note that columns longer than 255 characters will be truncated in the file.
- **DIF** The output is a file in the standard Data Interchange Format.

- **FIXED** The output is fixed format, with each column having a fixed width. The width for each column can be specified using the COLUMN WIDTH clause. If this clause is omitted, the width for each column is computed from the data type for the column, and is large enough to hold any value of that data type. No column headings are output in this format.
- **FOXPRO** The output is a FoxPro format file (the FoxPro memo field is different than the dBASE memo field) with the column definitions at the top of the file. Note that a maximum of 128 columns can be output. Also, note that columns longer than 255 characters will be truncated in the file.
- **LOTUS** The output is a Lotus WKS format worksheet. Column names will be put as the first row in the worksheet. Note that there are certain restrictions on the maximum size of Lotus WKS format worksheets that other software (such as Lotus 1-2-3) can load. There is no limit to the size of file Interactive SQL can produce.
- **SQL** The output is a Interactive SQL INPUT statement required to recreate the information in the table.
- **TEXT** The output is a TEXT format file, which prints the results in columns with the column names at the top and vertical lines separating the columns. This format is similar to that used to display data in the Interactive SQL data window.
- **WATFILE** The output is a WATFILE format file, with the column definitions at the top of the file.

OUTPUT_LENGTH option [ISQL

Function Controls the length used when Interactive SQL exports information to an external file.

Allowed values Integer

Default 0 (no truncation)

Description This option controls the length used when Interactive SQL exports information to an external file (using output redirection or the OUTPUT statement).

PERCENT_AS_COMMENT option

Function Controls the interpretation of the percent character.

Allowed values ON, OFF

Default ON

Description It is recommended that you not use % as a comment marker.

Versions of this product before Version 6 treated the percent character (%) in SQL statements exclusively as a comment delimiter. Since Version 5, alternative comment markers such as //, /* */, and -- (double dash) have been available. The double-dash style is the SQL/92 comment delimiter.

Adaptive Server Enterprise treats **%** as a modulo operator and does not support the Adaptive Server Anywhere **mod** function. Writing a statement that works in both environments and performs a modulo operation was previously impossible.

The PERCENT_AS_COMMENT option controls the meaning of %. The default setting for is ON, for backwards compatibility. You can set the option to OFF for compatibility with Adaptive Server Enterprise.

Procedures, triggers and views that were created with %-style comments are converted to double-dash comments when they are stored in the catalog.

Existing procedures must be recreated before changing option
Any existing procedures that contain %-style comments must be recreated before you change the option setting; otherwise, the procedures will fail to load.

The Sybase Central code editor does not highlight %-style comments. If you wish to have your comments highlighted in the Sybase Central editor, you should use one of the other comment delimiters.

PRECISION option

Function Specifies the maximum number of digits in the result of any decimal arithmetic.

Allowed Values Integer, with a maximum of 127

Scope Can be set for an individual connection or the PUBLIC group. Takes effect immediately.

Default 30

Description Precision is the total number of digits to the left and right of the decimal point. The "SCALE option" on page 172 specifies the minimum number of digits after the decimal point when an arithmetic result is truncated to the maximum PRECISION.

Multiplication, division, addition, subtraction, and aggregate functions can all have results that exceed the maximum precision.

For example, when a DECIMAL(8,2) is multiplied with a DECIMAL(9,2), the result could require a DECIMAL(17,4). If PRECISION is 15, only 15 digits will be kept in the result. If SCALE is 4, the result will be a DECIMAL(15,4). If SCALE is 2, the result will be a DECIMAL(15,2). In both cases, there is a possibility of overflow.

PREFETCH option

Function The PREFETCH option acts as a toggle allowing you to turn fetching on and off.

Allowed Values ON, OFF

Scope Can be set for an individual connection or the PUBLIC group. Takes effect immediately.

Default ON

Description This option controls whether rows are fetched to the client side in advance of being made available to the client application. Fetching a number of rows at a time, even when the client application requests rows one at a time (for example, when looping over the rows of a cursor) both cuts down on response time and improves overall throughput by cutting down the number of requests to the database.

The setting of PREFETCH is ignored by Open Client and JDBC connections.

QUERY_PLAN_ON_OPEN option

Function Controls whether a plan is returned when a cursor is opened.

Allowed values ON, OFF

Default OFF

Description In early versions of the software, each time an OPEN was done on a cursor, the server would return in the SQLCA **sqlerrmc** field a string representing the query plan (limited to 70 bytes). A more complete description can be obtained using the EXPLAIN statement or the PLAN function. For this reason, computing and returning the query plan on an OPEN is needed only for compatibility with old applications. The QUERY_PLAN_ON_OPEN option controls whether the plan is returned on an OPEN. By default, the setting is OFF.

QUOTED_IDENTIFIER option

Function Controls the interpretation of strings that are enclosed in double quotes.

Allowed values ON, OFF

Default ON

OFF for Open Client and JDBC connections

Description This option controls whether strings that are enclosed in double quotes are interpreted as identifiers (ON) or as literal strings (OFF). The QUOTED_IDENTIFIER option is included for Transact-SQL compatibility.

Sybase Central resents QUOTED_IDENTIFER temporarily to ON if it is set to OFF. A message is displayed informing you of this change. The change is in effect only for the Sybase Central connection.

☞ For more information, see "Setting options for Transact-SQL compatibility" on page 794 of the book *Adaptive Server Anywhere User's Guide*.

RECOVERY_TIME option

Function Sets the maximum length of time, in minutes, that the database server will take to recover from system failure.

Allowed Values Integer, in minutes

Scope Can be set only for the PUBLIC group. Takes effect when server is restarted.

Default 2

Description This option is used with the "CHECKPOINT_TIME option" on page 147 to decide when checkpoints should be done.

Adaptive Server Anywhere uses a heuristic to estimate the recovery time based on the operations that have been performed since the last checkpoint. Thus, the recovery time is not exact.

☞ For more information, see "Recovery from system failure" on page 569 of the book *Adaptive Server Anywhere User's Guide*.

ROW_COUNTS option

Function Specifies whether the database will always count the number of rows in a query when it is opened.

Allowed Values ON, OFF

Scope Can be set for an individual connection or the PUBLIC group. Takes effect immediately.

Default OFF

Description If this option is set to OFF, the row count is usually only an estimate. If this option is set to ON, the row count is always accurate. Opening queries may take significantly longer.

REPLICATE_ALL option

Function Allows an entire database to act as a primary site in a Replication Server setup.

Allowed values ON, OFF

Default OFF

Description This option is used by the LTM only. When it is set to ON, the entire database is set to act as a primary site in a Replication Server installation. All changes to the database are sent to Replication Server by the LTM.

REPLICATION_ERROR option

Function For SQL Remote, allows you to specify a stored procedure to be called by the Message Agent when a SQL error occurs.

Allowed values Stored procedure name.

Default No procedure

Description For SQL Remote, the REPLICATION_ERROR option allows you to specify a stored procedure to be called by the Message Agent when a SQL error occurs. By default no procedure is called.

The procedure must have a single argument of type CHAR, VARCHAR, or LONG VARCHAR. The procedure is called once with the SQL error message and once with the SQL statement that causes the eror.

Although the option allows you to track and monitor SQL errors in replication, you must still design them out of your setup; this option is not intended to resolve such errors.

RI_Trigger_time option

Function Controls the relative timing of referential integrity checks and trigger actions.

Allowed values BEFORE, AFTER

Default AFTER

Description The option can be set to either BEFORE or AFTER. When it's set to AFTER, referential integrity actions are executed after the UPDATE or DELETE.

Only the PUBLIC setting can be used; any other setting is ignored.

Prior to release 5.5, the referential integrity triggers were always fired BEFORE. Starting with release 5.5, the default has been AFTER.

The reason for the change of default behavior is that BEFORE triggers can lead to infinitely recursing triggers, when delete self-referencing rows.

SCALE option

Function Specifies the minimum number of digits after the decimal point when an arithmetic result is truncated to the maximum PRECISION.

Allowed Values Integer, with a maximum of 127.

Scope Can be set for an individual connection or the PUBLIC group. Takes effect immediately.

Default 6

Description Multiplication, division, addition, subtraction, and aggregate functions can all have results that exceed the maximum precision. See "PRECISION option" on page 168 for an example.

SQL_FLAGGER_ERROR_LEVEL option

Function Controls the response to any SQL that is not part of a specified set of SQL/92.

Allowed values E, I, F, or W

Default W

Description This option flags any SQL that is not part of a specified set of SQL/92 as an error.

The allowed values of *level* 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
- **W** Allow all supported syntax

SQL_FLAGGER_WARNING_LEVEL option

Function

Controls the response to any SQL that is not part of a specified set of SQL/92.

Allowed values

E, I, F, or W

Default

W

Description

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

The allowed values of *level* 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
- **W** Allow all supported syntax

STATISTICS option [ISQL

Function

Controls whether execution times, optimization strategies and other statistics are displayed in the statistics window.

Allowed values

0, 3, 4, 5, or 6.

Default

3

Description

When the option is set to 0, the statistics window is not displayed. Otherwise, the value represents the height of the statistics window, in lines.

STRING_RTRUNCATION option

Function

Determines whether an error is raised when an INSERT or UPDATE truncates a CHAR or VARCHAR string.

Allowed values ON, OFF

Default OFF

Description If the truncated characters consist only of spaces, no exception is raised. The setting of ON corresponds to ANSI/ISO SQL/92 behavior. When it is set to OFF, the exception is not raised and the character string is silently truncated.

SUBSCRIBE_BY_REMOTE option

Function Controls interpretation of NULL or empty-string SUBSCRIBE BY values.

Allowed values ON, OFF

Default ON

Description When the option is set to ON, operations from remote databases on rows with a SUBSCRIBE BY value that is NULL or an empty string assume that the remote user is subscribed to the row. When it is set to OFF, the remote user is assumed not to be subscribed to the row.

TEXTSIZE option

Function Controls the maximum size (in bytes) of text or image type data to be returned with a SELECT statement.

Allowed values Integer, in bytes.

Default 32765 bytes

Description The **@@textsize** global variable stores the current setting. To reset to the default size (32765 bytes), which is also the maximum value, use the statement:

```
set textsize 0
```

THREAD_COUNT option

Function Historical

Allowed Values Integer

Default Not applicable

Description This option is now ignored by the server. For Version 6 you should use the -gn server command-line option to set the maximum number of requests that the server handles simultaneously.

For information on the command-line switch, see "The database server" on page 12.

TIME_FORMAT option

Function Sets the format for times retrieved from the database.

Allowed Values A string composed of the symbols listed below.

Scope Can be set for an individual connection or the PUBLIC group. Takes effect immediately.

Default HH:NN:ss.SSS

For Open Client and JDBC connections the default is set to HH:NN:SS.SSS.

Description The format is a string using the following symbols:

- **hh** Two digit hours (24 hour clock)
- **nn** Two digit minutes
- **mm** Two digit minutes if following a colon (as in hh:mm)
- **ss[.s...]** Two digit seconds plus optional fraction

Each symbol is substituted with the appropriate data for the time that is being formatted. Any format symbol that represents character rather than digit output can be put in uppercase, which causes the substituted characters to also be in uppercase. For numbers, using mixed case in the format string suppresses leading zeros.

TIMESTAMP_FORMAT option

Function Sets the format for timestamps that are retrieved from the database.

Allowed Values A string composed of the symbols listed below.

Scope Can be set for an individual connection or the PUBLIC group. Takes effect immediately.

Default YYYY-MM-DD HH:NN:ss.SSS

For Open Client and JDBC connections the default is set to YYYY-MM-DD HH:NN:SS.SSS.

Description

The format is a string using the following symbols:

Symbol	Description
yy	Two digit year
Yyyy	Four digit year
Mm	Two digit month, or two digit minutes if following a colon(as in 'hh:mm')
Mmm[m...]	Character short form for months—as many characters as there are m's
dd	Two digit day of month
Ddd[d...]	Character short form for day of the week
hh	Two digit hours
nn	Two digit minutes
ss.ssssss	Seconds and fractions of a second, up to six decimal places. Not all platforms support timestamps to a precision of six places.
aa	am or pm (12 hour clock)
pp	Pm if needed (12 hour clock)
f	Use French days and months (deprecated)

Each symbol is substituted with the appropriate data for the timestamp that is being formatted. Any format symbol that represents character rather than digit output can be put in uppercase, which causes the substituted characters to also be in uppercase. For numbers, using mixed case in the format string suppresses leading zeros.

TRUNCATION_LENGTH option [ISQL

Function

Controls the truncation of wide columns for displays to fit on a screen.

Allowed values

Integer

Default

30

Description

When SELECT statement results are displayed on the screen, each column of output is limited to the width of the screen. The TRUNCATION_LENGTH option is used to reduce the width of wide columns so that more than one column will fit on the screen. A value of 0 means that columns are not truncated.

The default TRUNCATION_LENGTH is 30. For character-mode systems, this is an actual number of characters. For windowing systems, TRUNCATION_LENGTH is used to estimate an area of the screen to be used for display since proportional fonts are used.

TSQL_HEX_CONSTANT option

Function Controls whether hexadecimal constants are treated as binary typed constants.

Allowed values ON, OFF

Default ON

Description When this option is set to ON, hexadecimal constants are treated as binary typed constants. To get the historical behavior, set the option to OFF.

TSQL_VARIABLES option

Function Controls whether the @ sign can be used as a prefix for Embedded SQL host variable names.

Allowed values ON, OFF

Default OFF

ON for Open Client and JDBC connections

Description When this options set to ON, you can use the @ sign instead of the colon as a prefix for host variable names in Embedded SQL. This is implemented primarily for Transact-SQL compatibility.

VERIFY_ALL_COLUMNS option

Function Controls whether messages that contain updates published by the local database are sent with all column values included.

Allowed values ON, OFF

Default OFF

Description This option is used by SQL Remote only. When it is set to ON, messages that contain updates published by the local database are sent with all column values included, and a conflict in any column triggers a RESOLVE UPDATE trigger at the subscriber database.

VERIFY_THRESHOLD option

Function Controls which columns are verified when updates are replicated.

Allowed values Integer, in bytes

Default 1000

Description This option is used by SQL Remote only. If the data type of a column is longer than the threshold, old values for the column are not verified when an UPDATE is replicated. This keeps the size of SQL Remote messages down, but has the disadvantage that conflicting updates of long values are not detected.

WAIT_FOR_COMMIT option

Function Determines when foreign key integrity is checked, as data is manipulated.

Allowed Values ON or OFF

Scope Can be set for an individual connection or the PUBLIC group. Takes effect immediately.

Default OFF

Description If this option is set to ON, the database does not check foreign key integrity until the next COMMIT statement. Otherwise, all foreign keys that are not created with the CHECK ON COMMIT option are checked as they are inserted, updated or deleted.