SYBASE®

An SAP® Company

Utility Guide

# Sybase IQ 15.4

# Contents

# Audience

This guide is for Sybase® IQ utility program users who require reference material for the utility programs.

Utility programs are commands that you invoke directly from the operating system. Familiarity with relational database systems and introductory user-level experience with Sybase IQ is assumed. Use this book to get information about available syntax, parameters, and options. Other books in the Sybase IQ 15.4 documentation set provide more context on how to carry out particular tasks.

# CP874toUTF8 Database Administration Utility

The **CP874toUTF8** utility converts data in the CP874 character set into a UTF8 collation that is supported by Sybase IQ for the Thai language.

You can also use this utility to load data in the CP874 character set without converting it to UTF8.

### *Syntax*

```
CP874toUTF8 [CP874InputFile]
```

### *Usage*

You can run this utility only from the command prompt.

**CP874toUTF8** reads the named file in the CP874 character set (or standard input if no files are named) and prints the UTF8 conversion to standard output.

**Note:** Files with embedded NULL characters (‘\0’) are not converted correctly. Remove such characters before running this utility.

Use caution with large data files; the UTF8 output can be up to three times larger than the input data. Input and output file size must both be within operating system limits.

**CP874toUTF8** returns a 0 exit status upon successful completion. A nonzero exit status indicates an error occurred. The **CP874toUTF8** utility writes all error messages to stderr.

**CP874toUTF8** calls the International Components for Unicode (ICU) library to perform the data conversion. If ICU cannot convert the data, **CP874toUTF8** reports a conversion error. If the conversion fails, divide the file into smaller sections for conversion, to isolate the point of failure. If you cannot resolve the problem, contact Sybase Technical Support.

# dbbackup Database Administration Utility

The dbbackup utility makes a copy of the transaction log of a running IQ database and lets you truncate the transaction log, freeing disk space and improving recovery speed without having to stop and restart your server.

**Note:** To back up an entire Sybase IQ database, always use **BACKUP** instead of **dbbackup**. **BACKUP** backs up all database files, and is the only way to back up the catalog store. See *Reference: Statements and Options > SQL Statements > BACKUP statement*.

### *Syntax*

```
dbbackup [options] target-directory
```

### *Parameters*

This table lists the available options for the **dbbackup** utility.

**Table 1. dbbackup options**

| Option | Description |
|---|---|
| *@data* | Reads options from the specified environment variable or configuration file. If both exist, and share the same name, Sybase IQ uses the environment variable. For more information about configuration files, see the *Installation and Configuration Guide*.<br><br>To protect passwords or other information in the configuration file, use the File Hiding utility (**dbfhide**) to obfuscate configuration file contents. |
| **-c** "*keyword=value*; ... " | Supplies database connection parameters. If you do not specify the connection parameters, connection parameters from the SQL CONNECT environment variable are used, if set. The user ID must have DBA authority or REMOTE DBA authority. See *System Administration Guide: Volume 1 > Connection and Communication Parameters*. |

| Option | Description |
|---|---|
| **-l (lowercase L)** *file* | Sends a live backup of the transaction log to a file. Enables a secondary system to be brought up rapidly in the event of server failure. A live backup does not terminate, but continues while the server runs. It runs until the primary server becomes unavailable. At that point, it shuts down, but the backed up log file is intact and can be used to quickly bring up a secondary system.<br><br>The live backup of the transaction log is always the same length or shorter than the active transaction log. When a live backup is running and another backup restarts the transaction log (dbbackup **-x**), the live backup automatically truncates the live backup log and restarts the live backup at the beginning of the new transaction log.<br><br>See *System Administration Guide: Volume 1 > Data Backup, Recovery, and Archiving > How to Back Up Databases > Types of Backups > Making a Live Backup of a Transaction Log*. |
| **-o** | Writes output messages to the named file. |
| **-q** | Quiet mode — does not display output messages. This option is available only when you run this utility from a command prompt. |
| **-r** | Copies the old transaction log to a new name and starts a new empty log. The following steps occur:<br>• Sybase IQ copies and saves the current working transaction log to the directory specified in the command.<br>• Sybase IQ keeps the current transaction log in its current directory, but renames it using the format `yymmdd`*xx*`.log`, where *xx* are sequential characters starting at *AA* and running to *ZZ*, and *yymmdd* represents the current year, month, and day. This file is then no longer the current transaction log.<br>• Sybase IQ generates a new transaction log file that contains no transactions. The new file has the name of the former current transaction log and becomes the current transaction log. |
| **-t** | Backs up only the transaction log file. 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 files. |
| **-xo** *filename* | Truncate (delete and restart) the transaction log |

| Option | Description |
|---|---|
| **target-directory** | The directory to which the backup files are copied. If the directory does not exist, Sybase IQ creates it. The parent directory must exist. |

*Usage*

The **dbbackup** utility allows you to back up the transaction log while other applications or users are using the database. Backup file names are the same as the database file names.

The **dbbackup** utility truncates the database name to 70 characters and creates a target file with a truncated name. Sybase IQ uses **dbbackup** when synchronizing secondary servers. Due to `dbbackup` restrictions, database names must be less than 70 characters long.

If you have adequate disk space, use **-r** to preserve the existing log file under a new name and start a new empty log. If disk space is limited, use **-xo** instead to truncate the existing log.

Exit codes are 0 (success) or nonzero (failure).

*Examples*

The following Windows command backs up the transaction log from the `iqdemo` database running on the `sample_server` server into the directory `iqbackup`, connecting as user ID `DBA` with password `sql`:

```
dbbackup -c "eng=sample_server;dbn=iqdemo;uid=DBA;pwd=sql" c:\sample
\iqbackup
```

## See also

- *dbfhide Database Administration Utility* on page 9

# dbfhide Database Administration Utility

The **dbfhide** utility is a file hiding utility you use to add simple encryption to configuration files and initialization files to hide the contents of each file.

*Syntax*

```
dbfhide original-configuration-file encrypted-configuration-file
```

*Parameters*

This table lists the available options for the **dbfhide** utility.

| Option | Description |
|---|---|
| *original-configuration-file* | Name of the original file. |
| *encrypted-configuration-file* | Name for the new obfuscated file. |

*Usage*

Configuration files are used by some utilities to hold command-line options. These options can contain a password. You can use the **dbfhide** utility to add simple encryption to configuration files and `.ini` files used by Sybase IQ and its utilities, and thereby obfuscate the contents of the file. The original file is not modified. Once you add simple encryption to a file, there is no way to remove it. To change an obfuscated file, keep a copy of the original file that you can modify and obfuscate again.

*Examples*

Create a configuration file that starts the personal database server and the sample database. The file should set a cache of 10MB, and name this instance of the personal server *"Elora"*. The configuration file would be written as follows:

```
# Configuration file for server Elora -n Elora -c 10M path\asademo.db
```

Lines beginning with # are treated as comments.

Name the file `sample.txt`. To start the database using this configuration file, enter:

```
start_iq @sample.txt
```

Add simple encryption to the configuration.

```
dbfhide sample.txt encrypted_sample.txt
```

Use the encrypted_sample.txt file to start a database:

```
start_iq @encrypted_sample.txt
```

For more information about using configuration files, see the *Installation and Configuration Guide*.

The following command adds simple encryption to the `asaldap.ini` file:

```
dbfhide asaldap.ini encrypted_asaldap.ini
```

**See also**

- *dbbackup Database Administration Utility* on page 5

## Hiding the Contents of .ini Files Using dbfhide

Obfuscate an .ini file using the **dbfhide** utility.

1. Save the file with a different name.

   ```
   rename saldap.ini saldap.ini.org
   ```

   If you do not keep a copy of the original file, then you cannot modify the contents of the file once it has been obfuscated.
2. Obfuscate the file with the File Hiding utility, giving the obfuscated file the required file name:

   ```
   dbfhide saldap.ini.org saldap.ini
   ```
3. Protect the `saldap.ini.org` file using file system or operating system protection, or store the file in a secure location.

   To change the `saldap.ini` file, edit the `saldap.ini.org` file and repeat step 2.

   **Warning!** You should not add simple encryption to the `.odbc.ini` system information file with the File Hiding utility (dbfhide) on UNIX unless you will only be using Sybase IQ data sources. If you plan to use other data sources, then obfuscating the contents of the `.odbc.ini` file may prevent other drivers from functioning properly.

   dbfhide does not accept the **@data** parameter to read in options from a configuration file.

# dbinfo Database Administration Utility

The **dbinfo** utility displays information about a database catalog store. The information returned by **dbinfo** does not reflect the IQ store.

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

*Syntax*

```
dbinfo [ options ]
```

*Parameters*

This table lists the available options for the **dbinfo** utility.

**Table 2. dbinfo options**

| Option | Description |
|---|---|
| **-c** *"keyword=value; ..."* | Specify connection parameters. See *System Administration Guide: Volume 1 > Connection and Communication Parameters*. |
| **-o** *filename* | Write output messages to the named file. |
| **-q** | Operate quietly; do not display output messages. |
| **-u** | Output page usage statistics. Display information about the usage and size of all catalog store tables, including system and user-defined tables. You can only request page usage statistics if no other users are connected to the database. |

*Usage*

The `dbinfo` utility indicates when the database was created, the name of any transaction log file or log mirror that is maintained, the catalog store page size, the version of installed Java classes, and other information. Optionally, it can also provide catalog table usage statistics and details.

# dbisql Interactive SQL Utility

Interactive SQL (**dbisql**) is a graphical utility included with Sybase IQ that lets you execute SQL statements, build scripts, and display database data.

**Note:** For backward compatibility, Sybase IQ includes the older Interactive SQL Classic (**dbisqlc**) utility. Sybase recommends using Interactive SQL rather than Interactive SQL Classic. Interactive SQL Classic is deprecated and will be removed in a future release of Sybase IQ.

**See also**

- *Appendix: dbisqlc Interactive SQL Classic Utility (Deprecated)* on page 197
- *iqisql Interactive SQL Utility* on page 105
- *isql Interactive SQL Utility* on page 113

## Start Interactive SQL

You can start Interactive SQL from a command prompt, from the Windows **Start** menu, or from within Sybase Central.

### Starting Interactive SQL from a Command Prompt

Start Interactive SQL from the command prompt on Windows or UNIX.

1. Do one of the following:
   - In a command shell, enter `dbisql`
   - Select **Start** > **Run** and enter `dbisql`
2. In the **Connect** window, supply parameters.

**See also**

- *Interactive SQL Connection Parameters* on page 17
- *Interactive SQL Command Line Options* on page 15

### Starting Interactive SQL from the Windows Start Menu

Start Interactive SQL from the Windows Start menu.

1. Select **Start** > **Programs** > **Sybase** > **Sybase IQ 15.4** > **Interactive SQL**.
2. In the **Connect** window, supply parameters.

**See also**

- *Interactive SQL Connection Parameters* on page 17
- *Interactive SQL Command Line Options* on page 15

### Starting Interactive SQL from Sybase Central

Start Interactive SQL from Sybase Central.

1. In the left pane, select the Sybase IQ plug-in and do one of the following:
   - Choose **Tools > Sybase IQ 15 > Open Interactive SQL**.
   - Right-click a database, and choose **Open Interactive SQL**.
   - Right-click a stored procedure, and choose **Execute from Interactive SQL**.
2. In the **Connect** window, supply parameters.

**See also**

- *Interactive SQL Connection Parameters* on page 17
- *Interactive SQL Command Line Options* on page 15

## Interactive SQL Utility Syntax

Invoke Interactive SQL from a command prompt.

```
dbisql [ options ] [ dbisql-command | command-file ]
```

**Note:** Interactive SQL does not accept @filename parameters. Exit codes are 0 (success) or non-zero (failure).

- 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`. If there is an error in the command file, the process terminates.

```
dbisql -c "uid=DBA;pwd=sql" -onerror exit mycom.sql
```

- The following command, when entered on a single line at a command prompt, adds a user to the current default database:

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

**See also**

- *Interactive SQL Command Line Options* on page 15

## Interactive SQL Command Line Options

Specify options when invoking Interactive SQL from a command prompt.

**Table 3. Interactive SQL options**

| Option | Description |
|---|---|
| **-c** *"keyword=value; ..."* | Specifies connection parameters. See *System Administration Guide: Volume 1 > Connection and Communication Parameters* for a description of the connection parameters. If you do not specify this option, the environment variable `SQLCONNECT` is used. If Interactive SQL cannot connect, you see a dialog box where you can enter the connection parameters.<br><br>**Note:** Sybase recommends that you always specify connection parameters for Interactive SQL instead of relying on defaults. If you start more than one database on a server, for example, specify the database name, and in a network with subnets, specify the communications protocol parameter with host number. See *System Administration Guide: Volume 1 > Sybase IQ Connections*. |
| **-d** *delimiter* | Specifies a command delimiter. Quotation marks around the delimiter are optional, except when the command shell itself interprets the delimiter in some special way.<br><br>Command delimiters are used for all connections in that Interactive SQL session, regardless of the setting stored in the database (for the user, or the PUBLIC setting). |
| **-d1** | (The final character is the number 1, not a lower case L.) Echoes all statements that Interactive SQL executes to the Command window (STDOUT). This feedback is useful when debugging SQL scripts, or when Interactive SQL is processing a long SQL script |
| **-datasource** *dsn-name* | Specifies an ODBC data source to connect to. You do not need to be using the SQL Anywhere JDBC driver to use this option. However, if the data source to which you are connecting is not configured to use TCP/IP, you must use the SQL Anywhere JDBC driver to connect. By default, Sybase IQ data sources are configured to use TCP/IP. |
| **-f** file name | Opens (but does not run) the file called *file name*. You must enclose the file name in quotes if the file name contains a blank; otherwise, quotes are optional. If the file does not exist, or if it is a directory instead of a file, Interactive SQL prints an error message to the console and quits. If the file name does not include a full drive and path specification, the file is assumed to be relative to the current directory. |

| Option | Description |
|---|---|
| **-host** *host name* | Specifies the host name or IP address of the computer on which the database server is running. You can use the name **localhost** to represent the current machine. |
| **-nogui** | Runs Interactive SQL in a command-prompt mode, with no windowed user interface. This is useful for batch operations. If you specify either *dbisql-command* or *command-file*, then **-nogui** is assumed.<br><br>In **-nogui** mode, Interactive SQL sets the program exit code to indicate success or failure. On Windows operating systems, the environment variable ERRORLEVEL is set to the program exit code. Exit codes are:<br><br>• 0 — Success.<br>• 1 — General failure. At some point, a SQL or Interactive SQL statement did not execute successfully and the you chose to stop executing SQL statements. Alternatively, Interactive SQL noted an internal error.<br>• 5 — User terminated Interactive SQL. When an error occurs during execution, you are prompted to ignore the error, stop, or exit Interactive SQL. If you opt to exit, the program returns code 5. The program also returns code 5 if an error occurs and the Interactive SQL option **ON_ERROR** is set to EXIT.<br>• 9 — Unable to connect.<br>• 255 — Bad command. The command line contained incomplete or invalid switches.<br><br>**Note:** In **-nogui** mode, any SQL text you enter at the command prompt is executed when you press [**Enter**], even if you specify a command delimiter using the **-d** switch. Ensure the SQL command is complete before you press [**Enter**]. |
| **-onerror** (continue \| exit) | Controls what happens if an error is encountered while reading statements from a command file. This option overrides the **on_error** setting. This option is useful when using Interactive SQL in batch operations. |
| **-port** *portnumber* | Specifies the port number on which the database server is running. The default port number for Sybase IQ is 2638. |
| **-q** | Runs in quiet mode—does not display output messages. This option is useful only if you start Interactive SQL with a command or command file. Specifying this option does not suppress error messages. |
| **-version** | Displays the version number of Interactive SQL. |
| **-x** | Scans commands but does not execute them. You may find this option useful for checking long command files for syntax errors. |

**See also**

- *Interactive SQL Connection Parameters* on page 17
- *Starting Interactive SQL from Sybase Central* on page 14
- *Starting Interactive SQL from a Command Prompt* on page 13
- *Starting Interactive SQL from the Windows Start Menu* on page 13
- *Interactive SQL Utility Syntax* on page 14

### Setting Quiet Mode Prerequisites

If you use Interactive SQL with the **-q** option (quiet mode), and if the data extraction commands (primarily setting the option TEMP_EXTRACT_NAME1 to an output file) are in a command file, you must first set and make permanent the **Show all result sets** option.

If you do not set this option, the data extraction output file is not created.

1. Start Interactive SQL.
2. In the Interactive SQL window, select **Tools** > **Options**. The Options window appears.
3. Select **Sybase IQ** in the left pane.
4. Click the **Results** tab.
5. In the Results Processing area, select **Show all results sets**.
6. Click **OK**.

## Interactive SQL Connection Parameters

When no database is connected, Sybase IQ displays the Connect window to request connection parameters. Information you enter in the Connect window is not preserved between sessions.

The connection parameters you specify in the Connect window are dependent on the number of databases running on the database server. To connect to a single database, you only need to complete the **User ID** and **Password** fields if the IQ server was started on the local machine with *IQTMP15* environment variable set. If there are multiple databases running on the database server, you must specify additional connection parameters such as the server or database name.

Enter connection parameters in the Connect window using one of these methods:

- Using the Connect Assistant
- Manually

**Note:** You can bypass the Connect window by supplying the connection parameters on the command line with the **-c** option.

If the Connect window or an error message about missing information appears, you may need to enter the **-host** and **-port** or other missing information in the **Advanced** tab. If your database is on a remote server, enter the **-host** and **-port** parameters on separate lines, as in:

```
-host fiona
-port 1870
```

**See also**

- *Interactive SQL Command Line Options* on page 15
- *Starting Interactive SQL from Sybase Central* on page 14
- *Starting Interactive SQL from a Command Prompt* on page 13
- *Starting Interactive SQL from the Windows Start Menu* on page 13

## Supplying Connection Parameters Using the Connect Assistant

The Connect window has a Connect Assistant wizard to help you connect to a database. To display or hide the Connect Assistant, click the arrow in the top right corner of the window.

1. Click the **Next** button in the **Connect Assistant** area of the Connect window.
2. Follow the on-screen prompts.

**Note:** If you are connecting using an ODBC source, do not use the Connect Assistant. Close the Connect Assistant and click the **Identification** tab.

## Supplying Connection Parameters Manually

Connect to a database using the Identification, Database, Network, and Advanced Options tabs on the Connect window.

1. From the **Authentication** list, specify either **Database** or **Windows Integrated Login** authentication.
2. Select an **Action**.

**Table 4. Connect with an ODBC data source**

| Field/Button/Option | Details |
|---|---|
| ODBC data source name | Selects a data source, which is a stored set of connection parameters, for connecting to your database. This field is equivalent to the DSN connection parameter, which references a data source in the registry. |
| ODBC data source file | Selects a data source file for the connection. |

**Table 5. Connect to a running database on this computer**

| Field/Button/Option | Details |
|---|---|
| Server name | The name of the database server you are connecting to. For remote servers, specify the server as `host name:port number`. |
| Database name | The name of the database you are connecting to. You need a database name only if there is more than one database running on the database server. If the database is not already running on the server, you should specify the database file instead. You can also select a recently used database, or click Browse to locate the database file.<br>**Note:** If you specify both the database name and database file when trying to connect to a database that is already running, the database file is ignored. |

**Table 6. Connect to a running database on another computer**

| Field/Button/Option | Details |
|---|---|
| Host | The computer name on which the database server is running. |
| Port | The port the database server is using. |
| Server name | The name of the database server you are connecting to. For remote servers, specify the server as `host name:port number`. |
| Find | Looks for servers. |

**Table 7. Start and connect to a database on this computer**

| Field/Button/Option | Details |
|---|---|
| Database file | Sybase recommends that you type the full path and name of the database file. Otherwise, the path of the file is relative to the working directory of the database server. |
| Encryption key | If the database file is encrypted, you must supply an encryption key to the database server every time the database server starts the database. |

| Field/Button/Option | Details |
|---|---|
| Database name | The name of the database you are connecting to. You need a database name only if there is more than one database running on the database server. If the database is not already running on the server, you should specify the database file instead. You can also select a recently used database, or click Browse to locate the database file.<br><br>**Note:** If you specify both the database name and database file when trying to connect to a database that is already running, the database file is ignored. |
| Server name | The name of the database server you are connecting to. For remote servers, specify the server as `host name:port number`. |
| Start line | Enter a start line to connect to a local database server and to set your own start parameters. You must enter the full path of the database server. |
| Stop database after last disconnect | Automatically shut down the database after the last user disconnects. |

**Table 8. Start and connect to a database on another computer**

| Field/Button/Option | Details |
|---|---|
| Database file | Sybase recommends that you type the full path and name of the database file. Otherwise, the path of the file is relative to the working directory of the database server. |
| Encryption key | If the database file is encrypted, you must supply an encryption key to the database server every time the database server starts the database. |

| Field/Button/Option | Details |
|---|---|
| Database name | The name of the database you are connecting to. You need a database name only if there is more than one database running on the database server. If the database is not already running on the server, you should specify the database file instead. You can also select a recently used database, or click Browse to locate the database file.<br>**Note:** If you specify both the database name and database file when trying to connect to a database that is already running, the database file is ignored. |
| Server name | The name of the database server you are connecting to. For remote servers, specify the server as `host name:port number`. |
| Host | The computer name on which the database server is running. |
| Port | The port the database server is using. |
| Stop database after last disconnect | Automatically shut down the database after the last user disconnects. |

**Table 9. Connect with a connection string**

| Field/Button/Option | Details |
|---|---|
| Parameters | Connection string. |

3. Click **Connect**.

## Connect Window Tools

Click **Tools** at the bottom of the Connect window to access options.

**Table 10. Connect window tools**

| Tool | Description |
|---|---|
| Test connection | This tool tests whether the information provided results in a proper connection. |
| Copy connection string to clipboard | This tool creates a connection string from the options you specified in the Connect window and copies the string into your clipboard. |
| Save as ODBC Data Source | Saves the new data source to your list of data sources. |

## Execute SQL Statements and Command Files

After you execute a **SELECT** statement, the result set appears on the **Results** tab in the Results pane. By default, row numbers appear to the left of the result set.

## Using Text Completion

The text completion option supplies you with complete keywords and object names as you type.

For example, typing the letter S causes a text completion window to appear that includes a list of possible keywords or object names beginning with the letter S. You can configure the text completion settings from the Options window.

1. Type the first letter of a database object name in the **SQL Statements** pane.
2. By default, the text completion window opens automatically as you type. Alternatively, you can press **Ctrl**+**Space** or **Ctrl**+**Shift**+**Space** to open the window.
3. If you do not see the object name you want, press **Ctrl**+**A** to view a complete list of database objects (based on the filtering options you set—by default, all database objects appear in the list).
4. Select the object name from the list and then press **Enter**.

### Text Completion Keyboard Shortcuts

Keyboard shortcuts are available when the text completion list is open.

| Key | Description |
|---|---|
| Ctrl+A | Shows a context-free list of matches. |
| Ctrl+Double quote (") (or Ctrl+Shift+') | Completes the name, enclosing it in quotation marks, regardless of the setting of the quoted_indentifier option. |
| Ctrl+Asterisk (*) (or Ctrl+Shift+8) | For tables, inserts a comma-separated list of columns including data types. For stored procedures, inserts the procedure name, followed by a comma-separated list of parameter names and their data types. |
| Ctrl+C | Changes the contents of the text completion list to show or hide columns. |
| Ctrl+F | Changes the contents of the text completion list to show or hide SQL functions. |

| Key | Description |
|---|---|
| Ctrl+P | Changes the contents of the text completion list to show or hide stored procedures and functions. |
| Ctrl+Plus Sign (+) (or Ctrl+Shift+=) | For tables, inserts a comma-separated list of columns. For stored procedures, inserts the procedure name, followed by a comma-separated list of parameter names. |
| Ctrl+S | Changes the contents of the text completion list to show or hide system objects. |
| Ctrl+Shift+Space or Ctrl+Space | Opens the text completion window. You can also use Ctrl+Space to open the text completion window. |
| Ctrl+T | Changes the contents of the text completion list to show or hide tables. |
| Ctrl+V | Changes the contents of the text completion list to show or hide views. |
| Esc | Closes the text completion window without adding any text. |
| Tab | Accepts the selection and closes the text completion window. |

### See also

- *Interactive SQL Keyboard Shortcuts* on page 30

## Inserting Comments

Use comments to attach explanatory text to SQL statements or statement blocks.

You can turn text into a comment. The database server does not execute comments. Interactive SQL supports the following types of comments:

- -- (double hyphen)
- // (double slash)
- /* ... */ (slash-asterisk)

1. Select the text in the SQL Statements pane.
2. Press **Ctrl+Minus Sign (-)** to add double hyphen comment indicators or **Ctrl+Forward Slash (/)** to add double slash comment indicators.

   If no text is selected, the comment indicator is added to the beginning of the current line.

## Look Up Tables, Columns, and Procedures

While you are entering commands in Interactive SQL, you can look up the names of tables, columns, or procedures stored in the current database and insert them at your cursor position.

### Looking Up the Names of Tables in the Database

Look up the names of tables stored in the database and insert them at your cursor position.

1. Select **Tools > Lookup Table Name** or press F7.
2. Find and select the table.
3. Click **OK** to insert the table name into the SQL Statements pane at the current cursor position.

### Looking Up the Names of Columns in the Database

Look up the names of columns stored in the database and insert them at your cursor position.

1. Select **Tools > Lookup Table Name** or press F7.
2. Find and select the table containing the column.

   **Note:** In the Lookup Table Name window, you can enter the first few characters of the table you are looking for. The list is narrowed to include only those items that start with the text you entered.
3. Click **Show Columns**.
4. Select the column and click **OK** to insert the column name into the SQL Statements pane at the current cursor position.

### Looking Up the Names of Procedures in the Database

Look up the names of procedures stored in the database and insert them at your cursor position.

1. Select **Tools > Lookup Procedure Name** or press F8.
2. Find and select the procedure.

   **Note:** In the Lookup Procedure Name window, you can enter the first few characters of the procedure you are looking for. The list is narrowed to include only those items that start with the text you entered.
3. Click **OK** to insert the procedure name into the SQL Statements pane at the current cursor position.

## Indent SQL Statements

Increase or decrease indentation of SQL statements, or change the default number of spaces indented.

### Adding or Increasing Indentation of SQL Statements

Add indentation to an SQL statement or increase the indentation of an already-indented SQL statement.

1. Select the text in the SQL Statements pane that you want to indent. If no text is selected, the indentation is applied to the current line.
2. Press **Ctrl+Shift+Period**.

### Removing or Decreasing Indentation of SQL Statements

Remove indentation from an SQL statement or decrease the indentation of an already-indented SQL statement.

1. Select the text in the SQL Statements pane that you want to decrease the indentation. If no text is selected, the indentation is applied to the current line.
2. Press **Ctrl+Shift+Comma**.

### Changing the Number of Spaces that are Indented

Change the default indent size.

1. Select **Tools** > **Options**.
2. Choose **Editor** and then click the **Tabs** tab.
3. Type a new number in the **Indent Size** field.

## Clearing the SQL Statements Pane

Clear the SQL Statements pane when you no longer want to work with the SQL code displayed on the pane.

By default, the **Esc** key leaves the contents of the **SQL Statements** pane unchanged. Configure the **Esc** key to clear the SQL Statements pane and close any opened results sets.

1. Select **Tools** > **Options** > **Compatibility**.
2. Select **Pressing The Esc Key Clears SQL Statements And Closes Result Sets**.
3. Press **Esc**.

## Executing All SQL Statements

If multiple SQL statements exist in the SQL statements pane, you can execute all statements at once.

1. Type your query in the SQL Statements pane.
2. Press **F5**, or choose **SQL Execute** to execute the statement.

#### See also

- *Executing Selected SQL Statements* on page 26

- *Executing SQL Statements One at a Time* on page 26
- *Configuring the Execute Statements Toolbar Button* on page 26

## Executing Selected SQL Statements

If multiple SQL statements exist in the SQL statements pane, you can execute one or more selected statements.

1. Type your queries in the SQL Statements pane and select the query.
2. Press **F9**, or choose **SQL Execute Selection** to execute the statement.

#### See also

- *Executing All SQL Statements* on page 25
- *Executing SQL Statements One at a Time* on page 26
- *Configuring the Execute Statements Toolbar Button* on page 26

## Executing SQL Statements One at a Time

To execute SQL statements individually, use the **Single Step** option. This is useful when debugging.

1. Type your query in the SQL Statements pane.
2. Place your cursor in the statement that you want to execute.
3. Select **SQL > Single Step** or press **Shift+F9**.
   The selected SQL statement executes and the next SQL statement is selected.
4. Press **Shift+F9**.
   The selected SQL statement executes.
5. Repeat the previous step until there are no more selected statements to execute.

#### See also

- *Executing All SQL Statements* on page 25
- *Executing Selected SQL Statements* on page 26
- *Configuring the Execute Statements Toolbar Button* on page 26

## Configuring the Execute Statements Toolbar Button

Configure the **Execute Statements** button to either execute all SQL statements or only execute selected statements.

1. Select **Tools > Options > Toolbar**.
2. Perform one of these actions:
   - Select **Execute All Statement(s)**. This is the default setting.
   - Select **Execute Selected Statement(s)**.

**See also**

- *Executing All SQL Statements* on page 25
- *Executing Selected SQL Statements* on page 26
- *Executing SQL Statements One at a Time* on page 26

## Cancelling a SQL Statement

A cancel operation stops the current processing and prompts for the next command. The **Interrupt The SQL Statement** button on the Interactive SQL toolbar cancels a command. Click **Interrupt The SQL Statement** on the Interactive SQL toolbar.

**See also**

- *ON_ERROR Option [Interactive SQL]* on page 65

## Executing Command Files

Command files are text files that contain SQL statements and are useful if you want to run the same SQL statements repeatedly. You can use Interactive SQL to open, view, run, and save command files.
Execute the command file using one of these methods:

- Use the Interactive SQL **READ** statement to execute command files.

  For example:

  ```
  READ temp.sql;
  ```
- Load a command file into the SQL Statements pane and execute it directly from there.
- Load command files into the SQL Statements pane by choosing **File > Open**. Enter the file name when prompted.
- Run a command file without loading it by choosing **File > Run Script**.
- Supply a command file as a command line argument for Interactive SQL.

## Executing COMMIT and ROLLBACK Statements

There are multiple ways to execute **COMMIT** and **ROLLBACK** statements.

You can execute a **COMMIT** statement by:

- Choosing **SQL** > **Commit**.
- Pressing Ctrl+Shift+C.
- Typing `Commit` into the **SQL Statements** pane, and then executing the statement.

You can execute a **ROLLBACK** statement by:

- Choosing **SQL** > **Rollback**.
- Pressing Ctrl+Shift+R.
- Typing `Rollback` into the **SQL Statements** pane, and then executing the statement.

**Note:** Executing a **COMMIT** or **ROLLBACK** via the **SQL** menu or a keyboard shortcut does not modify the contents of the **SQL Statements** pane; however, the **Results** tab in the **Results** pane is cleared.

## Customizing the Interactive SQL Interface

You can configure settings for the tabs and panes in Interactive SQL using the Options window.

1. In Interactive SQL, choose **Tools** > **Options**.
2. In the left pane, click an option and specify the options that you want. You can change how results are displayed, specify whether or not to execute a commit after every statement (or only on exit or disconnect), and specify whether or not to check for updates.
3. Click **OK**.

## Interactive SQL Window Reference

Use the Interactive SQL window to enter SQL statements and view results and messages.

**Table 11. Interactive SQL window details**

| Pane | Column/Tab | Description |
|---|---|---|
| SQL Statements | | Provides a place for you to type SQL statements to access and modify your data. |
| | **Line Number** Column | A column on the left that shows line numbers. These line numbers allow you to do the following:<br>• Click a line number to select a line. Alternatively place your cursor in the line, and press **Ctrl+comma(,)**.<br>• Click and drag to select multiple lines.<br>• Double-click a line to select the entire SQL statement that corresponds to the line. Alternatively place your cursor in the statement, and press **Ctrl+period(.)**. |
| Results | | The Results pane has two tabs: **Results** and **Messages**. The tabs appear at the bottom of the Results pane. |

| Pane | Column/Tab | Description |
|---|---|---|
| | **Results** Tab | The **Results** tab displays the results of commands that you execute. For example, if you use SQL statements to search for specific data in the database, the **Results** tab displays the columns and rows that match the search criteria in the pane above. You can edit the result set on the **Results** tab. |
| | **Messages** Tab | The **Messages** tab displays messages from the database server about the SQL statements that you execute in Interactive SQL |

Results of graphical plans for IQ databases are displayed in separate Plan Viewer window(s).

### See also

- *View Plans Using the Interactive SQL Plan Viewer* on page 43

## Data Menu Window Reference

Use the windows available from the **Data** Menu to export a result set and import data.

**Table 12. Data menu windows**

| Window | Description |
|---|---|
| Export | Opens the Export Wizard, which allows you to export a result set. |
| Import | Opens the Import Wizard, which allows you to import data from a file or database. |

## Tools Menu Window Reference

Use the windows available from the **Tools** menu to configure Interactive SQL settings, search for table and procedure names to insert into your queries, and edit your queries.

**Table 13. Tools menu windows**

| Window | Description |
|---|---|
| Lookup Table Name | The Lookup Table Name window lets you browse table and column names and insert them into the SQL Statements pane. |

| Window | Description |
|---|---|
| Lookup Procedure Name | The Lookup Procedure Name window lets you browse procedure names and insert them into the SQL Statements pane. |
| Edit Query | The Query Editor provides a graphical way to create and edit SELECT statements in Interactive SQL. |
| Plan Viewer | The Plan Viewer is a graphical tool for viewing graphical plans for IQ databases. |
| Options | The Options window sets options for commands, appearance, importing and exporting data, and messages in Interactive SQL. |

### See also

- *View Plans Using the Interactive SQL Plan Viewer* on page 43
- *Creating a Query Using the Query Editor* on page 41

## Interactive SQL Keyboard Shortcuts

Learn about the keyboard shortcuts available in Interactive SQL.

**Table 14. Interactive SQL Keyboard Shortcuts**

| Function key | Description |
|---|---|
| Tab | Accepts the selection and closes the text completion window. |
| Alt+F4 | Exits Interactive SQL. |
| Ctrl+A | Shows a content-free list of matches. |
| Ctrl+C | Copies the selected rows and column headings to the clipboard in the **Results** pane.<br><br>In the **SQL Statements** pane, copies the selected text to the clipboard. |
| Ctrl+End | Moves to the bottom of the current pane. |
| Ctrl+H | Displays the history of your executed SQL statements. |
| Ctrl+Home | Moves to the top of the current pane. |
| Ctrl+N | Clears the contents of the Interactive SQL window. |
| Ctrl+P | Prints the contents of the **SQL Statements** pane. |

| Function key | Description |
|---|---|
| Ctrl+Q | Displays the Query Editor.<br>The Query Editor helps you build SQL queries. When you have finished building your query, click OK to export it back into the **SQL Statements** pane. |
| Ctrl+S | Saves the contents of the **SQL Statements** pane. |
| Ctrl+Plus Sign (+) | Adds the item with its parameter list to the **SQL Statements** pane. |
| Ctrl+Asterisk | Adds the item with its parameter and type list to the **SQL Statements** pane. |
| Ctrl+Double quote (") | Adds the item enclosed in quotation marks to the **SQL Statements** pane. |
| F2 | Edits the selected value in the result set. |
| F5 | Executes all text in the **SQL Statements** pane. |
| F7 | Displays the Lookup Table Name dialog. |
| F8 | Displays the Lookup Procedure Name dialog. |
| F9 | Executes the text that is selected in the **SQL Statements** pane.<br>If no text is selected, all of the statements are executed. |
| Pgdn | Moves a page down in the current pane. |
| Pgup | Moves a page up in the current pane. |

These keyboard shortcuts are available when the **SQL Statements** pane has the focus:

**Table 15. Interactive SQL Keyboard Shortcuts for SQL Statements Pane**

| Function key | Description |
|---|---|
| Ctrl+] | Moves the cursor to the matching parenthesis, braces, brackets, and angle brackets. |
| Ctrl+Backspace | Deletes the word to the left of the cursor. |
| Ctrl+Del | Deletes the word to the right of the cursor. |
| Ctrl+G | Opens the Go To dialog where you can specify the line you want to go to. |
| Ctrl+L | Deletes the current line from the **SQL Statements** pane and puts the line onto the clipboard. |
| Ctrl+Shift+] | Extends the selection to the matching brace. Brace matching matches parentheses, braces, brackets, and angle brackets. |
| Ctrl+Shift+C | Executes a **COMMIT** statement. |
| Ctrl+Shift+L | Deletes the current line. |

| Function key | Description |
|---|---|
| Ctrl+Shift+R | Executes a **ROLLBACK** statement. |
| Ctrl+Shift+U | Changes the selection to uppercase characters. |
| Ctrl+U | Changes the selection to lowercase characters. |
| F3 | Finds the next occurrence of the selected text. |
| Home | Moves the cursor to the start of the current line or to the first word on the current line. |
| Shift+F3 | Finds the previous occurrence of the selected text. |
| Shift+Home | Extends the selection to the start of the text on the current line. |

## Result Sets

Work with the results of commands that you execute.

### Showing Multiple Results Sets (UNIX)

By default, Interactive SQL shows the first result set of the most-recently executed statement. Use this procedure to see all result sets on UNIX platforms.

1. If running Interactive SQL as a command line program (-nogui mode):
   a) Navigate to your $HOME directory and locate the file `.isqlPreferences11`.
   b) Change the line`<entry key="SybaseIQ.showMultipleResultSets">0</entry>` to `<entry key="SybaseIQ.showMultipleResultSets">1</entry>`
   c) Save your changes to `.isqlPreferences11`.
2. If using GUI mode:
   a) Click **Tools** > **Options** in the Interactive SQL window.
   The Interactive SQL Options window appears.
   b) On the **Sybase IQ** page, select the **Results** tab.
   c) Choose **Show All Result Sets**.
   d) Click **OK**.

### Showing Multiple Result Sets (Windows)

By default, Interactive SQL shows the first result set of the most-recently executed statement. Use this procedure to see all result sets on Windows platforms.

1. Click **Tools** > **Options** in the Interactive SQL window.
   The Interactive SQL Options window appears.

2. On the **Sybase IQ** page, select the **Results** tab.
3. Choose **Show all Result Sets**.
4. Click **OK**.

## Edit Result Sets in Interactive SQL

Once you execute a query in Interactive SQL, you can sort and edit the result set to modify the database. You can also select rows from the result set and copy them for use in other applications.

Interactive SQL supports editing, inserting, and deleting rows. Editing the result set has the same effect as executing **UPDATE**, **INSERT**, and **DELETE** statements. After editing a result set, the equivalent **INSERT**, **UPDATE**, and **DELETE** statements are added to Interactive SQL's command history.

To edit a row or value in the result set, you must have the proper permissions on the table or column you want to modify values from. For example, if you want to delete a row, then you must have **DELETE** permission for the table the row belongs to.

You cannot edit a result set if you:

- Select columns from a table with a primary key, but do not select all the primary key columns
- Attempt to edit the result set of a **JOIN** (for example, if there is data from more than one table in the result set).
- Attempt to edit a table that has its editing disabled

Editing the result set may fail if you:

- Attempt to edit a row or column you do not have permission on.
- Enter an invalid value (for example, a string in a numeric column or a NULL in a column that does not allow NULLs).

When editing fails, an Interactive SQL error message appears explaining the error, and the database table values remain unchanged.

### See also

- *Recalling a Command* on page 40

### Editing Table Values from the Interactive SQL Result Set

You can change any or all of the values within existing rows in database tables, provided that you have UPDATE permission on the columns being modified. In addition table editing must not be disabled.

1. Execute a query in Interactive SQL.
2. On the **Results** tab, click the value you want to change.

3. Right-click the value and choose **Edit Row**, or press F2 to edit the result set.

   A blinking cursor appears in the table cell containing the value.
4. Enter the new value. If you want to change other values in the row, press **Tab** or **Shift+Tab** to move to the other values.
5. Press **Enter** to update the database once you are done editing values in the row.

   You can press the **Esc** key to cancel the change that was made to the selected value.
6. Execute a **COMMIT** statement to make your changes to the table permanent.

#### See also

- *Disabling Table Editing* on page 34

### Disabling Table Editing

You can disable table editing via the Options window in Interactive SQL.

1. From the **Tools** menu, choose **Options**, and then choose **Sybase IQ**.
2. Ensure that **Scrollable Table** is selected and select **Disable Editing**.
3. Click **OK**.
4. Execute a query. You must execute a new query for the changes to table editing to take effect.

#### See also

- *Editing Table Values from the Interactive SQL Result Set* on page 33

### Insert Rows into the Database from the Interactive SQL Result Set

Interactive SQL allows you to add new rows to a table. You tab between columns in the result set to add values to the row. You must have INSERT permission on the table to add new rows.

#### *Inserting a New Row into the Result Set*

Add a new blank row to the table from the result set.

1. Right-click the result set and choose **Add Row**. A new blank row appears with a blinking cursor in the first value in the row.
2. Enter the new value and then press **Tab** to move to the next column.

   You cannot enter invalid data types into a column. For example, you cannot enter a string into a column that accepts the INT data type. Repeat this step until all the column values are added.
3. Press **Enter** to update the database.

#### *Inserting Values into Columns with Default Values*

When adding a value in a column that has a default value, the cell editor contains a list with a (DEFAULT) item. Similarly, if a column accepts NULL values, (NULL) appears in the list. Select **(DEFAULT)** if you want to insert the default value.

If a column cannot be NULL and does not have a default value, you must enter a value.

#### *When the Result Set Contains a Computed Column*

If the result set contains a computed column and you do not specify a value for the computed column, the value is calculated when the database is updated. However, if you specify a value for the computed column, the database is updated with the specified value, and a value is not calculated for the computed column.

#### *Inserting New Rows Using the INPUT Statement*

An alternative to inserting new rows from the result set in Interactive SQL is to add rows using the **INPUT** statement with the **PROMPT** clause.

1. From Interactive SQL, add a row using the **INPUT** statement with the **PROMPT** clause. For example:

   ```
   INPUT INTO Products PROMPT;
   ```
2. Specify the value for each column when prompted.

### Deleting Rows from the Database Using Interactive SQL

You can also delete rows from a database table in Interactive SQL. You must have DELETE permission on the table to delete rows.

1. Select the row(s) you want to delete using one of these methods:
   a) Press and hold the **Shift** key while clicking the row(s).
   b) Press and hold the **Shift** key while using the Up or Down Arrow.
2. Press **Delete**.
3. Execute a **COMMIT** to make the change permanent.

### Copying Rows from an Interactive SQL Result Set

You can copy rows directly from the result set in Interactive SQL and then paste them into other applications.

1. Right-click any cell in the row and choose **Copy Data > Rows**. To copy multiple rows, hold the **Ctrl** key while clicking cells in the rows, and then right-click and choose**Copy Data > Rows**.
2. Paste the row(s) into other applications.

### Copying Columns from the Interactive SQL Result Set

You can copy columns directly from the result set in Interactive SQL and then paste them into other applications.

1. Right-click any cell in the column and choose **Copy Data > Columns**. To copy multiple columns, hold the **Ctrl** key while clicking cells in the columns, and then right-click and choose**Copy Data > Columns**.
2. Paste the columns(s) into other applications.

### Copying Cells from the Interactive SQL Result Set

Copying an individual value copies only the data to the clipboard. No column headings are copied and no quoting is done.

1. Right-click the cell you want to copy and choose **Copy Data > Cells**. To copy multiple cells, hold the **Ctrl** key while clicking cells, and then right-click and choose**Copy Data > Cells**.
2. Paste the cell(s) into other applications.

### Sorting Columns in an Interactive SQL Result Set

Sort the results by a specified column.

1. Click a column-header in the **Results** tab.
2. Fetch the remaining results if prompted.

## Generating SQL Statements from Result Sets

You can create **INSERT**, **DELETE**, and **UPDATE** statements for selected rows in the result set.

1. Select the row(s) you want to generate a statement for.
2. Right-click the selection, choose **Generate**, and then choose **INSERT Statement**, **DELETE Statement**, or **UPDATE Statement**. The statement is copied to the clipboard.

## Printing SQL Statements and Result Sets

You can print the contents of the SQL Statements pane or query results.

1. Type your query in the SQL Statements pane and execute the query, if desired.
2. Press **Ctrl+P** or select **File > Print**.
3. When prompted, choose to print the SQL statements or results.

### Adding a Header

You can add a header to printed SQL statements or printed query results.

1. Select **Tools > Options**. The Interactive SQL Options window appears.

2. On the Editor page, click the **Print** tab.
3. In the **Header** field, specify the text that you want to appear in the header. You can also click the right arrow and choose items to include in the header.

## Favorites List

You can add the current database connection and open SQL file to your Favorites list.

### Adding a .sql File to Favorites

Store frequently-used SQL command files in a favorites list. The favorites list is specific to a single user and cannot be seen by other users.

1. Open the SQL command file that you want to add to your favorites.
2. Select **Favorites > Add to Favorites**
3. Select **Add the open file 'filename'**. In the **Name** field, type a name for the **.sql** file.
4. Click **OK**.

#### See also

- *Favorites Menu Window Reference* on page 39

### Adding SQL Statements to Favorites

Store frequently-used SQL statements in a favorites list. The favorites list is specific to a single user and cannot be seen by other users.

1. Type the SQL commands that you want to add to your favorites in the **SQL Statements** pane. The contents in the SQL Statements pane cannot exceed 16384 characters.
2. From the **Favorites** menu, choose **Add To Favorites**.
3. Select **Add SQL Statements**. In the **Name** field, type a name for the favorite.
4. Click **OK**.

### Adding a Connection to Favorites

Store frequently-used connections in a favorites list. The favorites list is specific to a single user and cannot be seen by other users.

1. Connect to a database.
2. Select **Favorites > Add To Favorites**
3. Select **Save The Connection Password**. In the Name field, type a name for the connection.
4. Click **OK**.

**See also**

- *Favorites Menu Window Reference* on page 39

## Showing the Favorites

Show the list of frequently-used SQL command files and connections.
Select **Favorites > Show Favorites**.
The Favorites pane appears on the left side of the Interactive SQL window.

**See also**

- *Favorites Menu Window Reference* on page 39

## Opening a Favorite

Open an SQL command file or connections from your list of favorites.
From the **Favorites** menu, choose the favorite you want to open.

**See also**

- *Favorites Menu Window Reference* on page 39

## Editing Favorites

Edit the favorites list.

1. From the **Favorites** menu, choose **Show Favorites**.
2. Select a favorite, right-click, and then choose **Edit**.
3. Follow the instructions in the window.
4. Click **Save**.

## Exporting Favorites

Export all of your favorites to a `.fav` file.

1. Choose **Favorites** > **Export Favorites**.
2. Specify a file name for the `.fav` file and then click **Export**.

## Importing Favorites

Import favorites on to another computer or save the file as a backup.

1. Choose **Favorites** > **Import Favorites**.
2. Browse to the `filename.fav` file and then click **Import**.

## Favorites Menu Window Reference

Use the windows available from the Favorites menu to manage SQL files and connection information as favorites.

**Table 16. Favorite menu windows**

| Window | Description |
|---|---|
| Add to Favorites | Allows you to save SQL files and connection information as favorites. |
| Organize Favorites | Allows you to maintain and organize your favorites. |
| Show Favorites | Opens the Favorites window on the left side of the Interactive SQL window. |

**See also**

- *Adding a .sql File to Favorites* on page 37
- *Adding a Connection to Favorites* on page 37
- *Showing the Favorites* on page 38
- *Opening a Favorite* on page 38

# Command Recall

When you execute a command, Interactive SQL automatically saves it in a history list that persists between Interactive SQL sessions. You can also log your executed commands to a log file.

Interactive SQL maintains a record of up to 50 of the most recent commands.

You can view the entire list of commands in the Command History window. The most recent commands appear at the bottom of the list.

## Accessing the Command History Window

Open the Command History window to view the record of up to 50 of the most recent commands.
Use one of the following methods:

- Press **Ctrl+H**.
- Click the **Open A List Of Past SQL Statements** button on the toolbar.

## Recalling a Command

Recall commands you executed from the command history.

1. Open the Command History window and select the command.
2. Click **OK**.

**Note:** You can also recall commands without the Command History window. Use the **Recall Previous SQL Statement** and **Recall Next SQL Statement** icons in the toolbar to scroll back and forward through your commands, or press **Alt+Right Arrow** and **Alt+Left Arrow**, respectively.

### See also

- *Edit Result Sets in Interactive SQL* on page 33

## Copying Commands from the Command History Window

Copy commands from the Command History Window to the SQL Statements pane of Interactive SQL.

1. Open the Command History window.
2. Select the command or commands, and then press **Ctrl+C** or click **Copy**.
3. Click **OK**.
   The selected statements copy to the SQL Statements pane.

## Saving Commands from the Command History Window

Save the command history as a .sql file.

1. Open the Command History window.
2. Click the **Save History As .SQL File** button or press **Ctrl+S**.
3. In the Save As window, specify a location and name for the file. The command history file has a **.sql** extension.
4. Click **Save** when finished.

## Removing Commands from the Command History Window

Clear a single command, multiple commands, or all commands from the command history.

1. Open the Command History window.
2. Remove the commands using one of these methods:

   - Select one or more commands and click the **Delete** button. This action cannot be undone.

- Click **Clear History** to remove all the commands from the window. This action cannot be undone.

### Logging Commands

You can record commands as you execute them. The recorded commands are stored in a log file so you can use the commands again.

1. Select **SQL > Start Logging**.
2. In the Save As window, specify a location and name for the log file. For example, name the file `mylogs.sql`.
3. Click **Save** when finished.
4. Select **SQL > Stop Logging**.

## Build Queries with the Query Editor

The Query Editor is a tool in Interactive SQL that helps you build **SELECT** statements.

You can create SQL queries in the Query Editor, or you can import queries and edit them. When you have finished your query, click OK to export it back into Sybase Central or Interactive SQL for processing.

You do not need to use SQL code to create queries with the Query Editor. However, you can use SQL with the Query Editor in the following ways:

- You can create a query in the SQL Statements pane in Interactive SQL and import it into the Query Editor by highlighting the code before you open the editor.
- At any time while using the Query Editor, you can click SQL at the bottom of the window to see the SQL code for the query you are building. You can directly edit the code, and the fields are automatically updated in the Query Editor.

### Creating a Query Using the Query Editor

Use the Query Editor tools and tabs to write an SQL query.

1. Connect to a database from Interactive SQL.
2. Select **Tools > Edit Query**.

   If you have SQL code selected in Interactive SQL, the selected code is automatically imported into the Query Editor
3. Create your query.
4. Click **OK** to write the query to the SQL Statements pane.
5. Use the tabs that guide you through the components of a SQL query:

**Table 17. Query Editor Tabs**

| Tab | Description |
|---|---|
| Tables tab | Specifies the tables in your query. |
| Joins tab | Specifies a join strategy for combining the data in the tables. If you include more than one table in your query, you should specify a join strategy for combining the data in the tables. If you do not specify a join strategy for tables you added in the Tables tab, the Query Editor suggests one; if there is a foreign key relationship between the tables, it generates a join condition based on that relationship, or it suggests a cross product. When you open queries, the Query Editor accepts exactly the join strategy that you specified (and an unspecified JOIN is not defaulted to KEY JOIN). |
| Columns tab | Specifies the columns in your result set. If you do not specify columns, all columns appear.. |
| INTO tab | Assign results to variables. |
| WHERE tab | Specifies conditions for restricting the rows in your result set. |
| GROUP BY tab | Group rows in the result set. |
| HAVING tab | Restricts the rows in your result set based on group values. |
| ORDER BY tab | Sorts the rows. |

**6.** Use the following tools:

**Table 18. Query Editor Tools**

| Tool | Description |
|---|---|
| Expression Editor | Use the Expression Editor to build search conditions or define computed columns. |
| Derived Table | Use this window, which is nearly identical to the main Query Editor, to create derived tables and subqueries. |

## See also

- *View Plans Using the Interactive SQL Plan Viewer* on page 43
- *Tools Menu Window Reference* on page 29

### Configuring the Query Editor

Configure the Query Editor to customize settings.

1. Select **Tools** > **Options** > **SQL Anywhere**
2. Click the **Query Editor** tab and change settings.

### Query Editor Limitations

Learn about capabilities and syntax not supported by the Query Editor.

The Query Editor builds SQL Anywhere **SELECT** statements. It is not designed to create views, although you can create them in Interactive SQL and reference them in the Query Editor. Nor was it designed to create **UPDATE** statements or other non-SELECT SQL statements. It creates a single **SELECT** statement, so it does not build unions or intersects of **SELECT** statements. In addition, the Query Editor does not support Transact-SQL syntax.

## View Plans Using the Interactive SQL Plan Viewer

The Plan Viewer is a graphical tool for viewing graphical plans for databases.

The Plan Viewer window is divided into panes:

**Table 19. Plan Viewer panes**

| Pane | Description |
|---|---|
| SQL pane | Provides a place for you to type SQL statements that you want to generate plans for. |
| Results pane | Shows the graphical plan. |
| Details pane | Provides text details about the plan |

**See also**

- *Interactive SQL Window Reference* on page 28
- *Tools Menu Window Reference* on page 29
- *Creating a Query Using the Query Editor* on page 41

### Starting the Plan Viewer

Starting the Plan Viewer opens it in a separate window.

1. Open Interactive SQL.
2. Choose **Tools** > **Plan Viewer** (or press **Shift+F5**).

**See also**

- *Configuring the Graphical Plan* on page 44
- *Printing the Plan* on page 45
- *Viewing Graphical Plans in Interactive SQL* on page 44

## Viewing Graphical Plans in Interactive SQL

You can view the query optimizer's execution plan for an SQL statement in the Plan Viewer window in Interactive SQL.

1. Type your query in the SQL Statements pane.
2. Select **Tools > Plan Viewer** or press **Shift+F5**.

   The Plan Viewer appears in a separate window. Your specified query appears in the SQL pane.
3. Click **Get Plan** to generate a plan for the specified query.
4. Choose **Tools >  Plan Viewer**.
5. Click **Open**.
6. Select a plan file (.saplan), and then click **Open**.

**See also**

- *Configuring the Graphical Plan* on page 44
- *Printing the Plan* on page 45
- *Starting the Plan Viewer* on page 43

## Configuring the Graphical Plan

After executing the graphical plan, you can customize the appearance of items in the plan.

1. Right-click the plan in the lower left pane of the Plan Viewer and choose **Customize**.
2. Change the settings.
3. Click **OK** when finished.
4. Click **Get Plan** to generate the graphical plan with your changes.

**See also**

- *Printing the Plan* on page 45
- *Starting the Plan Viewer* on page 43
- *Viewing Graphical Plans in Interactive SQL* on page 44

### Printing the Plan

You can print a plan in the Plan Viewer.

1. Select **Tools** > **Plan Viewer**.
2. Press the **Print** button or right-click the plan and select **Print**.

#### See also

- *Configuring the Graphical Plan* on page 44
- *Starting the Plan Viewer* on page 43
- *Viewing Graphical Plans in Interactive SQL* on page 44

## Source Control Integration

Interactive SQL integrates with third-party source control systems, allowing you to perform many common source control operations on files from within Interactive SQL.

On Windows, Interactive SQL integrates with most source control products that support the Microsoft Common Source Code Control API (SCC), including Microsoft Visual SourceSafe. To use source control products that do not support the SCC API on Windows and other operating systems, specify a command line to run for each of the source control actions. Output from those commands appears in a log window.

Interactive SQL supports the following tasks (as long as the task is supported in the source control product):

- Open a source control project
- Get
- Check in
- Check out
- Undo check out
- Compare versions
- Show file history
- Show file properties
- Run the source control manager

If the underlying source control program does not support an action, its corresponding menu item is disabled. For example, Visual SourceSafe supports all of these actions, but using a custom (command line) source control system does not support opening a source control project, or running a source control manager.

You should be familiar with the operations of your source control program before attempting to use it from Interactive SQL.

## Configuring Interactive SQL to Use Source Control

You must configure Interactive SQL to use source control before you can perform source control actions on files, such as checking files in and out, comparing different versions of a file, and viewing the history for a file.

If you are running Interactive SQL on a Windows computer that has a source control product that supports the Microsoft SCC API, you can use that product or use a custom (command line oriented) system.

### Configuring Interactive SQL Source Control on Windows

Enable source control integration if you are running Interactive SQL on a Windows computer that has a source control product that supports the Microsoft SCC API.

1. Select **Tools** > **Options**.
2. In the left pane, click **Source Control**.
3. Click **Enable Source Control Integration**.
4. Click **OK**.

### Configuring Interactive SQL Source Control Systems with a Command Line Interface

Enable source control integration if you are running Interactive SQL on a computer that has a custom command line oriented source control product.

1. Click **Tools** > **Options**.
2. In the left pane, click **Source Control**.
3. Click **Enable Source Control Integration**.
4. Click **Configure**.
5. In the Custom Source Control Options window, click **Reset**.
6. Select your source control system from the list, and then click **OK**.
7. Edit the commands in the list as necessary by selecting an action from the Source Control Actions list, and then typing the corresponding command in the Command Line pane.

   When you are defining commands for your system in the Source Control Actions list, use the placeholder `[FILENAME]` to represent the name of the file that is used when you run the command.

   If you do not specify a command line for an action, the item in the **File** > **Source Control** menu is disabled.
8. Click **OK**, and then click **OK** again.

## Opening Source Control Projects from Interactive SQL

Some source control products require you to open a source control project before you can perform any other source control actions.

The exact definition of what a project is depends on the source control system you are using. Typically, it is a set of files that are under source control, along with a location on your local file system where working copies of the files are placed. You usually have to provide some credentials, such as a user ID and password, to the source control system to open a project.

If your source control system supports opening a source control project, the **File > Source Control > Open Source Control Project** menu item is enabled. Choosing this option from the **File** menu opens a source control-specific window for opening a project. Once you open a project, you do not have to open it again, even in subsequent Interactive SQL sessions. The project is opened automatically for you.

### See also

- *Additional Source Control Actions* on page 48
- *Checking Out Files from Interactive SQL* on page 47
- *Checking In Files from Interactive SQL* on page 48

## Checking Out Files from Interactive SQL

Once you open a file in Interactive SQL, use the command on the **File** menu to check the file out.

1. Choose **File > Open** and then browse to the file you want to open.

   The file status (Checked In, Checked Out, or Not Controlled) appears on the status bar at the bottom of the Interactive SQL window.
2. Check out the file by choosing **File > Source Control > Check Out**.
3. Depending on which source control product you are using, you may be prompted for a comment or other options as part of the check out procedure.

   **Note:** If you are using a SCC-compliant source control system, the status is always accurate. However, if you use the custom source control system, the status is based on whether the file is read-only or not. A read-only file is assumed to be checked in, but no assumptions are made about editable files because they could be either checked out or not controlled.

### See also

- *Opening Source Control Projects from Interactive SQL* on page 47
- *Additional Source Control Actions* on page 48

## Checking In Files from Interactive SQL

When you are finished making edits to your file, you can check it back in from Interactive SQL.

1. Choose **File > Source Control > Check In**.
2. Enter check in comments if you are prompted.

### See also

- *Opening Source Control Projects from Interactive SQL* on page 47
- *Additional Source Control Actions* on page 48

## Additional Source Control Actions

In addition to opening source control projects, and checking files in and out, Interactive SQL supports several other source control actions. The availability of these actions depends on the source control system you are using.

Access these actions from the **File > Source Control** menu in Interactive SQL.

**Table 20. Additional source control actions**

| Action | Description |
|---|---|
| Get | Gets the latest copy of the file you currently have open in the SQL Statements pane. |
| Undo Check Out | Discards your working copy of the file, and then downloads the copy of the file that is in the source control archive. |
| Compare Versions | This action compares the working copy of the file you have opened against the version in the source control archive. |
| History | Displays a list of source control actions (typically check-ins) that have been made to the file you have open. |
| Properties | Displays a list of source control properties that are associated with the file you have opened. |
| Run Source Control Manager | Launches the management program for your source control system. For example, if you are using Microsoft Visual SourceSafe, this launches Visual SourceSafe Explorer. |

### See also

- *Opening Source Control Projects from Interactive SQL* on page 47

- *Checking Out Files from Interactive SQL* on page 47
- *Checking In Files from Interactive SQL* on page 48

## SQL Statements for Interactive SQL

Interactive SQL statements can only be used from within Interactive SQL.

### CLEAR Statement [Interactive SQL]

Closes any open result sets in Interactive SQL (**dbisql**).

#### Syntax

```
CLEAR
```

#### Usage

Closes any open result sets and leaves the contents of the SQL Statements pane unchanged

Side effects:

The **CLEAR** statement closes the cursor associated with the data being cleared.

#### Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not applicable.

#### Permissions

None

### CONFIGURE Statement [Interactive SQL]

Activates the Interactive SQL (**dbisql**) configuration window.

#### Syntax

```
CONFIGURE
```

#### Usage

The **dbisql** configuration window displays the current settings of all **dbisql** options. It does not display or let you modify database options.

If you select Permanent, the options are written to the `SYSOPTION` table in the database and the database server performs an automatic **COMMIT**. If you do not choose Permanent, and instead click OK, options are set temporarily and remain in effect for the current database connection only.

### Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not supported by Adaptive Server Enterprise.

### Permissions

None

## CONNECT Statement [ESQL] [Interactive SQL]

Establishes a connection to a database.

### Syntax

Syntax 1

```
CONNECT
… [ TO engine-name ]
…[ DATABASE database-name ]
…[ AS connection-name ]
…[ USER ] userid [ IDENTIFIED BY ]
```

Syntax 2

```
CONNECT USING connect-string
```

### Parameters

- **engine-name: –** identifier, string, or host-variable
- **database-name: –** identifier, string, or host-variable
- **connection-name: –** identifier, string, or host-variable
- **userid: –** identifier, string, or host-variable
- **password: –** identifier, string, or host-variable
- **connect-string: –** a valid connection string or host-variable

### Examples

- **Example 1 –** **CONNECT** usage within Embedded SQL:

  ```
  EXEC SQL CONNECT AS :conn_name
  USER :userid IDENTIFIED BY :password;
  EXEC SQL CONNECT USER "dba" IDENTIFIED BY "sql";
  ```

- **Example 2 – CONNECT** usage from **dbisql**:
  - Connect to a database from **dbisql**. Prompts display for user ID and password:

```
CONNECT
```

- Connect to the default database as DBA, from **dbisql**. A password prompt displays:

```
CONNECT USER "DBA"
```

- Connect to the demo database as the DBA, from **dbisql**:

```
CONNECT
TO <machine>_iqdemo
USER "DBA"
IDENTIFIED BY sql
```

where *<machine>_iqdemo* is the engine name.

- Connect to the demo database using a connect string, from **dbisql**:

```
CONNECT
USING 'UID=DBA;PWD=sql;DBN=iqdemo'
```

## Usage

The **CONNECT** statement establishes a connection to the database identified by *database-name* running on the server identified by *engine-name*.

Embedded SQL behavior—In Embedded SQL, if no *engine-name* is specified, the default local database server is assumed (the first database server started). If no *database-name* is specified, the first database on the given server is assumed.

The **WHENEVER** statement, **SET SQLCA**, and some **DECLARE** statements do not generate code and thus might appear before the **CONNECT** statement in the source file. Otherwise, no statements are allowed until a successful **CONNECT** statement has been executed.

The user ID and password are used for permission checks on all dynamic SQL statements. By default, the password is case-sensitive; the user ID is not.

For a detailed description of the connection algorithm, see *System Administration Guide: Volume 1 > Sybase IQ Connections > How Sybase IQ Establishes Connections*.

DBISQL behavior—If no database or server is specified in the **CONNECT** statement, **dbisql** remains connected to the current database, rather than to the default server and database. If a database name is specified without a server name, **dbisql** attempts to connect to the specified database on the current server. You must specify the database name defined in the **-n** database switch, not the database file name. If a server name is specified without a database name, **dbisql** connects to the default database on the specified server. For example, if this batch is executed while connected to a database, the two tables are created in the same database.

```
CREATE TABLE t1( c1 int );
CONNECT DBA IDENTIFIED BY sql;
CREATE TABLE t2 (c1 int );
```

No other database statements are allowed until a successful **CONNECT** statement has been executed.

The user ID and password are used for checking the permissions on SQL statements. If the password or the user ID and password are not specified, the user is prompted to type the missing information. By default, the password is case-sensitive; the user ID is not.

Multiple connections are managed through the concept of a current connection. After a successful connect statement, the new connection becomes the current one. To switch to a different connection, use **SET CONNECTION**. Executing a **CONNECT** statement does not close the existing connection (if any). Use **DISCONNECT** to drop connections.

Static SQL statements use the user ID and password specified with the *-l* option on the **SQLPP** statement line. If no *-l* option is given, then the user ID and password of the **CONNECT** statement are used for static SQL statements also.

Connecting with no password—If you are connected to a user ID with DBA authority, you can connect to another user ID without specifying a password. (The output of **dbtran** requires this capability.) For example, if you are connected to a database from Interactive SQL as DBA, you can connect without a password with the statement:

```
CONNECT other_user_id
```

In Embedded SQL, you can connect without a password by using a host variable for the password and setting the value of the host variable to be the null pointer.

AS clause—connection can optionally be named by specifying the **AS** clause. This allows multiple connections to the same database, or multiple connections to the same or different database servers, all simultaneously. Each connection has its own associated transaction. You might even get locking conflicts between your transactions if, for example, you try to modify the same record in the same database from two different connections.

Syntax 2—A *connect-string* is a list of parameter settings of the form **keyword**=*value*, and must be enclosed in single quotes.

### Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Open Client Embedded SQL supports a different syntax for the **CONNECT** statement.

### Permissions

None

## DISCONNECT Statement [Interactive SQL]

Drops a connection with the database.

### Syntax

```
DISCONNECT [ { connection-name | CURRENT | ALL } ]
```

### Parameters

- **connection-name: –** identifier, string, or host-variable

### Examples

- **Example 1 –** How to use **DISCONNECT** in Embedded SQL:

  ```
  EXEC SQL DISCONNECT :conn_name
  ```

- **Example 2 –** How to use **DISCONNECT** from **dbisql** to disconnect all connections:

  ```
  DISCONNECT ALL
  ```

### Usage

The **DISCONNECT** statement drops a connection with the database server and releases all resources used by it. If the connection to be dropped was named on the **CONNECT** statement, then the name can be specified. Specifying **ALL** drops all of the connections of the application to all database environments. **CURRENT** is the default and drops the current connection.

An implicit **ROLLBACK** is executed on connections that are dropped.

### Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—Supported by Open Client/Open Server.

### Permissions

None

## OUTPUT Statement [Interactive SQL]

Writes the current query results to a file.

### Syntax

```
OUTPUT TO filename
[ APPEND ] [ VERBOSE ]
[ FORMAT output-format ]
[ ESCAPE CHARACTER character ]
[ DELIMITED BY string ]
[ QUOTE string [ ALL ] ]
[ COLUMN WIDTHS ( integer, … ) ]
[ HEXADECIMAL { ON | OFF | ASIS } ]
[ ENCODING encoding ]
[ WITH COLUMN NAMES   ]
```

## Parameters

- **output-format: – TEXT | FIXED | HTML | SQL | XML**
- **encoding: –** *string* or *identifier*

## Examples

- **Example 1 –** Place the contents of the `Employees` table in a text file:

  ```
  SELECT * FROM Employees;
  OUTPUT TO employees.txt FORMAT TEXT
  ```

- **Example 2 –** Place the contents of the `Employees` table at the end of an existing file, and include any messages about the query in this file as well:

  ```
  SELECT * FROM Employees;
  OUTPUT TO employees.txt APPEND VERBOSE
  ```

- **Example 3 –** Export a value that contains an embedded line feed character. A line feed character has the numeric value 10, which you can represent as the string '\x0a' in a SQL statement.

  Execute this statement with **HEXADECIMAL ON**:

  ```
  SELECT 'line1\x0aline2'; OUTPUT TO file.txt HEXADECIMAL ON
  ```

  The result is a file with one line in it, containing this text:

  ```
  line10x0aline2
  ```

  Execute the same statement with **HEXADECIMAL OFF**:

  ```
  line1\x0aline2
  ```

  If you set `HEXADECIMAL` to **ASIS**, you get a file with two lines:

  ```
  'line1
  line2'
  ```

  Using **ASIS** generates two lines, because the embedded line feed character has been exported without being converted to a two-digit hex representation, and without a prefix.

## Usage

The **OUTPUT** statement copies the information retrieved by the current query to a file.

You can specify the output format with the optional **FORMAT** clause. If no **FORMAT** clause is specified, the Interactive SQL `OUTPUT_FORMAT` option setting is used.

The current query is the **SELECT** or **LOAD TABLE** statement that generated the information that appears on the Results tab in the Results pane. The **OUTPUT** statement reports an error if there is no current query.

**Note: OUTPUT** is especially useful in making the results of a query or report available to another application, but is not recommended for bulk operations. For high-volume data movement, use the **ASCII** and **BINARY** data extraction functionality with the **SELECT** statement. The extraction functionality provides much better performance for large-scale data movement, and creates an output file you can use for loads.

APPEND—This optional keyword is used to append the results of the query to the end of an existing output file without overwriting the previous contents of the file. If the **APPEND** clause is not used, the **OUTPUT** statement overwrites the contents of the output file by default. The **APPEND** keyword is valid if the output format is TEXT, FIXED, or SQL.

VERBOSE—When the optional **VERBOSE** keyword is included, error messages about the query, the SQL statement used to select the data, and the data itself are written to the output file. If **VERBOSE** is omitted (the default), only the data is written to the file. The **VERBOSE** keyword is valid if the output format is TEXT, FIXED, or SQL.

FORMAT—Allowable output formats are:

- TEXT—The output is a TEXT 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, all values (not just strings) are quoted. TEXT is the default output format.

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

  If you are exporting Java methods that have string return values, you must use the **HEXADECIMAL OFF** clause.
- FIXED—The output is fixed format with each column having a fixed width. The width for each column can be specified using the **COLUMN WIDTHS** clause. No column headings are output in this format.

  If **COLUMN WIDTHS** 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. The exception is that `LONG VARCHAR` and `LONG BINARY` data defaults to 32KB.
- HTML—The output is in the Hyper Text Markup Language format.
- SQL—The output is an Interactive SQL **INPUT** statement required to recreate the information in the table.

  **Note:** Sybase IQ does not support the **INPUT** statement. You need to edit this statement to a valid **LOAD TABLE** (or **INSERT**) statement to use it to load data back in.
- XML—The output is an XML file encoded in UTF-8 and containing an embedded DTD. Binary values are encoded in CDATA blocks with the binary data rendered as 2-hex-digit strings. The **LOAD TABLE** statement does not accept XML as a file format.

ESCAPE CHARACTER—The default escape character for characters stored as hexadecimal codes and symbols is a backslash (\), so \x0A is the line feed character, for example.

This default can be changed using the **ESCAPE CHARACTER** clause. For example, to use the exclamation mark as the escape character, enter:

```
... ESCAPE CHARACTER '!'
```

DELIMITED BY—The **DELIMITED BY** clause is for the TEXT output format only. The delimiter string is placed between columns (default comma).

QUOTE—The **QUOTE** clause is for the TEXT output format only. The quote string is placed around string values. The default is a single quote character. If ALL is specified in the **QUOTE** clause, the quote string is placed around all values, not just around strings.

COLUMN WIDTHS—The **COLUMN WIDTHS** clause is used to specify the column widths for the **FIXED** format output.

HEXADECIMAL—The **HEXADECIMAL** clause specifies how binary data is to be unloaded for the TEXT format only. When set to **ON**, binary data is unloaded in the format **0xabcd**. When set to **OFF**, binary data is escaped when unloaded (**\xab\xcd**). When set to **ASIS**, values are written as is, that is, without any escaping—even if the value contains control characters. ASIS is useful for text that contains formatting characters such as tabs or carriage returns.

ENCODING—Specifies the encoding that is used to write the file. The **ENCODING** clause can be used only with the TEXT format.

If *encoding* is not specified, Interactive SQL determines the code page that is used to write the file as follows, where code page values occurring earlier in the list take precedence over those occurring later:

- The code page specified with the **DEFAULT_ISQL_ENCODING** option (if this option is set)
- The default code page for the computer Interactive SQL is running on

Side Effects

- In Interactive SQL, the Results tab displays only the results of the current query. All previous query results are replaced with the current query results.

### Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not applicable.

### Permissions

None

## PARAMETERS Statement [Interactive SQL]

Specifies parameters to an Interactive SQL (**dbisql**) command file.

### Syntax

```
PARAMETERS parameter1, parameter2, …
```

### Examples

- **Example 1 –** This **dbisql** command file takes two parameters:

```
PARAMETERS department_id, file ;
SELECT Surname
FROM Employees
WHERE DepartmentID = {department_id}
>#{file}.dat;
```

### Usage

**PARAMETERS** specifies how many parameters there are to a command file and also names those parameters so that they can be referenced later in the command file.

Parameters are referenced by putting the named parameter into the command file where you want the parameter to be substituted:

```
{parameter1}
```

There must be no spaces between the braces and the parameter name.

If a command file is invoked with fewer than the required number of parameters, **dbisql** prompts for values of the missing parameters.

### Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not applicable.

### Permissions

None

## READ Statement [Interactive SQL]

Reads Interactive SQL (**dbisql**) statements from a file.

### Syntax

```
READ filename [ parameter ] …
```

### Examples

- **Example 1 –**

```
READ status.rpt '160'
```

- **Example 2 –**

```
READ birthday.sql [>= '1988-1-1'] [<= '1988-1-30']
```

- **Example 3 –**

```
[test1.sql]
PARAMETERS par1, par2;
```

```
BEGIN
DECLARE v_par1 int;
DECLARE v_par2 varchar(200);

SET v_par1 = {par1};
SET v_par2 = {par2};

MESSAGE STRING('PAR1 Value: ', v_par1 ) TO CLIENT;
MESSAGE STRING('PAR2 Value: ', v_par2 ) TO CLIENT;

END;
```

```
(USR1)> READ test1.sql 123 '041028'
PAR1 Value: 123
PAR2 Value: 041028
```

**Note:** The second parameter value 041028 must be enclosed in quotes, as *v_par2* is declared as a character data type.

## Usage

The **READ** statement reads a sequence of **dbisql** statements from the named file. This file can contain any valid **dbisql** statement, including other **READ** statements, which can be nested to any depth.

To find the command file, **dbisql** first searches the current directory, then the directories specified in the environment variable SQLPATH, then the directories specified in the environment variable PATH. If the named file has no file extension, **dbisql** also searches each directory for the same file name with the extension `.sql`.

Parameters can be listed after the name of the command file. These parameters correspond to the parameters named on the **PARAMETERS** statement at the beginning of the statement file (see *PARAMETERS Statement* ). **dbisql** then substitutes the corresponding parameter wherever the source file contains:

```
{ parameter-name }
```

where *parameter-name* is the name of the appropriate parameter.

The parameters passed to a command file can be identifiers, numbers, quoted identifiers, or strings. When quotes are used around a parameter, the quotes are put into the text during the substitution. Parameters that are not identifiers, numbers, or strings (contain spaces or tabs) must be enclosed in square brackets (**[ ]**). This allows for arbitrary textual substitution in the command file.

SQL character literals, including character data, passed as parameters to a **READ** statement should be enclosed in quotes.

If not enough parameters are passed to the command file, **dbisql** prompts for values for the missing parameters.

The **READ** statement also supports an **ENCODING** clause, which lets you specify the encoding that is used to read the file. See *SQL Anywhere Server - SQL Reference > SQL statements > SQL statements > READ statement [Interactive SQL]*.

**Note:** This reference points to SQL Anywhere documentation.

### Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not applicable.

### Permissions

None

## SET CONNECTION Statement [ESQL] [Interactive SQL]

Changes the active database connection.

### Syntax

```
SET CONNECTION [connection-name]
```

### Parameters

- **connection-name: –** identifier, string, or host-variable

### Examples

- **Example 1 –** In Embedded SQL:

  ```
  EXEC SQL SET CONNECTION :conn_name
  ```

- **Example 2 –** From **dbisql**, set the current connection to the connection named “conn1”:

  ```
  SET CONNECTION conn1
  ```

### Usage

The current connection state is saved and is resumed when it again becomes the active connection. If *connection-name* is omitted and there is a connection that was not named, that connection becomes the active connection.

**Note:** When cursors are opened in Embedded SQL, they are associated with the current connection. When the connection is changed, the cursor names are not accessible. The cursors remain active and in position and become accessible when the associated connection becomes active again.

### Standards

- SQL— **dbisql** use is a vendor extension to ISO/ANSI SQL grammar. Embedded SQL is a full-level feature.
- Sybase—Supported by Open Client/Open Server.

### Permissions

None

## SET OPTION Statement [Interactive SQL]

Changes Interactive SQL (**dbisql**) options.

### Syntax

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

### Usage

**SET PERMANENT** (Syntax 2) stores all current **dbisql** options in the `SYSOPTION` system table. These settings are automatically established every time **dbisql** is started for the current user ID.

Syntax 3 is used to display all of the current option settings. If there are temporary options set for **dbisql** or the database server, these display; otherwise, permanent option settings are displayed.

If you incorrectly type the name of an option when you are setting the option, the incorrect name is saved in the `SYSOPTION` table. You can remove the incorrectly typed name from the `SYSOPTION` table by setting the option PUBLIC with an equality after the option name and no value:

```
SET OPTION PUBLIC.a_mistyped_name=;
```

## START DATABASE Statement [Interactive SQL]

Starts a database on the specified database server.

### Syntax

```
START DATABASE database-file
… [ AS database-name ]
… [ ON engine-name ]
… [ AUTOSTOP { YES | NO } ]
… [ KEY key ]
```

### Examples

- **Example 1 –** On a UNIX system, start the database file `/s1/sybase/sample_2.db` on the current server:

  ```
  START DATABASE '/s1/sybase/sample_2.db'
  ```

- **Example 2 –** On a Windows system, start the database file `c:\sybase\sample_2.db` as `sam2` on the server `eng1`:

  ```
  START DATABASE 'c:\sybase\sample_2.db'
  AS sam2
  ON eng1
  ```

### Usage

The database server must be running. The full path must be specified for the database file unless the file is located in the current directory.

The **START DATABASE** statement does not connect **dbisql** to the specified database: a **CONNECT** statement must be issued to make a connection.

If *database-name* is not specified, a default name is assigned to the database. This default name is the root of the database file. For example, a database in file `c:\sybase\IQ_15\demo\iqdemo.db` is given the default name `iqdemo`.

If *engine-name* is not specified, the default database server is assumed. The default database server is the first started server among those currently running.

The default setting for the **AUTOSTOP** clause is YES. With **AUTOSTOP** set to YES, the database is unloaded when the last connection to it is dropped. If **AUTOSTOP** is set to NO, the database is not unloaded.

If the database is strongly encrypted, enter the KEY value (password) using the **KEY** clause.

Sybase recommends that you start only one database on a given Sybase IQ database server.

### Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not applicable.

### Permissions

Must have DBA authority.

## START ENGINE Statement [Interactive SQL]

Starts a database server.

### Syntax

```
START ENGINE AS engine-name [ STARTLINE command-string ]
```

### Examples

- **Example 1 –** Start a database server named `eng1` without starting any databases on it:

  ```
  START ENGINE AS eng1
  ```

- **Example 2 –** Use of the **STARTLINE** clause:

  ```
  START ENGINE AS eng1 STARTLINE 'start_iq -c 8096'
  ```

### Usage

To specify a set of options for the server, use the **STARTLINE** keyword together with a command string.

Valid command strings are those that conform to the database server command line description in *Utility Guide > start_iq Database Server Startup Utility*.

**Note:** Several server options are required for Sybase IQ to operate well. To ensure that you are using the right set of options, Sybase recommends that you start your server by using either Sybase Central or a configuration file with the **start_iq** command.

### Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not applicable.

### Permissions

None

## STOP DATABASE Statement [Interactive SQL]

Stops a database on the specified database server.

### Syntax

```
STOP DATABASE database-name
… [ ON engine-name ]
… [ UNCONDITIONALLY ]
```

### Examples

- **Example 1 –** Stop the database named `sample` on the default server:

  ```
  STOP DATABASE sample
  ```

### Usage

If *engine-name* is not specified, all running engines are searched for a database of the specified name.

The *database-name* is the name specified in the **-n** parameter when the database is started, or specified in the **DBN** (**DatabaseName**) connection parameter. This name is typically the file name of the database file that holds the catalog store, without the `.db` extension, but can be any user-defined name.

If **UNCONDITIONALLY** is supplied, the database is stopped, even if there are connections to the database. If **UNCONDITIONALLY** is not specified, the database is not stopped if there are connections to it.

### Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not applicable.

### Permissions

Must have DBA authority.

## STOP ENGINE Statement [Interactive SQL]

Stops a database server.

### Syntax

```
STOP ENGINE engine-name [ UNCONDITIONALLY ]
```

### Examples

- **Example 1 –** Stop the database server named `sample`:

  ```
  STOP ENGINE sample
  ```

### Usage

If **UNCONDITIONALLY** is supplied, the database server is stopped, even if there are connections to the server. If **UNCONDITIONALLY** is not specified, the database server is not stopped if there are connections to it.

### Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not applicable.

### Permissions

None

## SQL Options for Interactive SQL

Interactive SQL options are a subset of the IQ SQL options. These change how Interactive SQL interacts with the database.

### DEFAULT_ISQL_ENCODING Option [Interactive SQL]

Specifies the code page used by **READ** and **OUTPUT** statements.

*Allowed Values*
*identifier* or *string*

*Default*
Use system code page (empty string)

*Scope*
Can only be set as a temporary option, for the duration of the current connection.

*Description*
`DEFAULT_ISQL_ENCODING` is used to specify the code page to use when reading or writing files. It cannot be set permanently. The default code page is the default code page for the platform you are running on. On English Windows machines, the default code page is 1252.

Interactive SQL determines the code page that is used for a particular **OUTPUT** or **READ** statement as follows, where code page values occurring earlier in the list take precedence over those occurring later in the list:

- The code page specified in the **ENCODING** clause of the **OUTPUT** or **READ** statement
- The code page specified with the `DEFAULT_ISQL_ENCODING` option (if this option is set)
- The default code page for the computer on which Interactive SQL is running

For a list of supported code pages, see *SQL Anywhere Server – Database Administration > Configuring Your Database > International languages and character sets > Character set and collation reference information > Supported and alternate collations*.

See also *SQL Anywhere Server – Database Administration > Configuring Your Database > International languages and character sets > Understanding character sets > Understanding character sets*.

**Note:** These references point to SQL Anywhere documentation.

*Example*
Set the encoding to UTF-16 (for reading Unicode files):

```
SET TEMPORARY OPTION DEFAULT_ISQL_ENCODING = 'UTF-16'
```

## ON_ERROR Option [Interactive SQL]

Controls the action taken if an error is encountered while executing statements in Interactive SQL.

*Allowed Values*
String. See *Description* for allowed values.

*Default*
PROMPT

*Description*
Controls the action taken, if an error is encountered while executing statements:

- STOP – Interactive SQL stops executing statements from the file and returns to the statement window for input.
- PROMPT – Interactive SQL prompts the user to see if he or she wants to continue.
- CONTINUE – Errors display in the Messages pane and Interactive SQL continues executing statements.
- EXIT – Interactive SQL terminates.
- NOTIFY_CONTINUE – The error is reported, and the user is prompted to press **Enter** or click **OK** to continue.

- NOTIFY_STOP – The error is reported, and the user is prompted to press **Enter** or click **OK** to stop executing statements.
- NOTIFY_EXIT – The error is reported, and the user is prompted to press **Enter** or click **OK** to terminate Interactive SQL.

When you are executing a `.SQL` file, the values STOP and EXIT are equivalent.

### See also

- *Cancelling a SQL Statement* on page 27

# dblocate Database Administration Utility

The **dblocate** utility is a server location utility that assists in diagnosing connection problems by locating databases on the immediate TCP/IP network.

### *Syntax*

```
dblocate [ options ]
```

### *Parameters*

This table lists the options available for the **dblocate** utility.

**Table 21. dblocate options**

| Option | Description |
|---|---|
| **-d** | Lists the server name and address, for each server found, followed by a comma-separated list of databases running on that server. If the list exceeds 160 characters, it is truncated and ends with an ellipsis (...). |
| **-dn** database-name | Lists the server name and address, for servers running a database with the specified name. If the list exceeds 160 characters, it is truncated and ends with an ellipsis (...). |
| -dv | Displays the server name and address, for each server found, listing each database running on that server on a separate line. The list is not truncated, so this option can be used to reveal lists that are truncated when the v option is used. |
| -n | Lists IP addresses in the output, rather than computer names. This may improve performance since looking up computer names may be slow. |
| **-o** *filename* | Writes output messages to the named file. |
| **-p** *portnumber* | Displays the server name and address only for servers using the specified TCP/IP port number. The TCP/IP port number must be between 1 and 65535. |
| **-q** | Runs in quiet mode—messages are not displayed. |
| **-s** *name* | Displays the server name and address only for servers with the specified server name. If this option is used, the -ss option should not be used (if both options are used, it is likely that no matching servers will be found). |

| Option | Description |
|---|---|
| **-ss** *substr* | Displays the server name and address only for servers that contain the specified substring anywhere in the server name. If this option is used, the -s option should not be used (if both options are used, it is likely that no matching servers will be found). |
| -v | Displays the full server name. By default, dblocate truncates database server names that are longer than 40 bytes. |

*Usage*

**dblocate** locates any SQL Anywhere or Sybase IQ database servers running over TCP/IP on the immediate network. It prints a list of database servers and their addresses.

Depending on your network, the utility may take several seconds before printing its results.

### See also

- *start_iq Database Options* on page 184

# dblog Database Administration Utility

The **dblog** utility is a transaction log utility that displays or changes the name of the transaction log or transaction log mirror associated with your database.

You can also use **dblog** to stop a database from maintaining a transaction log mirror, or start maintaining a transaction log mirror.

Sybase IQ automatically handles the creation and deletion of the transaction log for a database. The database must run with a transaction log. The Sybase IQ server will not start without a transaction log. A transaction log mirror is a duplicate copy of a transaction log, maintained in tandem by the database. A transaction log mirror is not required, but Sybase recommends that you use one, especially if you do not frequently back up your IQ database.

*Syntax*

```
dblog [options] database-file
```

*Parameters*

This table lists the options available for the **dblog** utility.

**Table 22. dblog options**

| Option | Description |
|---|---|
| **-ek** *key* | Specify the encryption key for strongly encrypted databases directly in the command. If you have a strongly encrypted database, you must provide the encryption key to use the database or transaction log in any way. For strongly encrypted databases, you must specify either -ek or -ep, but not both. The command will fail if you do not specify the correct key for a strongly encrypted database. |
| **-ep** | Specify that you want to be prompted for the encryption key. This option causes a window to appear, in which you enter the encryption key. It provides an extra measure of security by never allowing the encryption key to be seen in clear text. For strongly encrypted databases, you must specify either -ek or -ep, but not both. The command will fail if you do not specify the correct key for a strongly encrypted database. |
| **-g** n | 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. |

| Option | Description |
|---|---|
| **-il** | Use this option if you have stopped using the Log Transfer Manager to participate in a Replication Server installation on this database, but continue to use SQL Remote or MobiLink synchronization. It resets the Log Transfer Manager log offset that is kept for the delete_old_logs option, allowing transaction logs to be deleted when they are no longer needed. |
| **-is** | Use this option if you have stopped using MobiLink synchronization on this database, but continue to use the Log Transfer Manager or SQL Remote. It resets the MobiLink log offset that is kept for the delete_old_logs option, allowing transaction logs to be deleted when they are no longer needed. |
| **-m** *mirror-name* | Specify the file name 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. |
| **-n** | Stop using a transaction log, and stop using a transaction log mirror. Without a transaction log, the database can no longer participate in data replication or use the transaction log in data recovery. If a SQL Remote, Log Transfer Manager, or dbmlsync truncation offset exists, the transaction log cannot be removed unless the corresponding ignore option (-il for the Log Transfer Manager, -ir for SQL Remote, or -is for dbmlsync) is also specified. You cannot stop using a transaction log if the database has auditing turned on (unless you first turn auditing off). |
| **-o** *filename* | Write output messages to the named file. |
| **-q** | Run in quiet mode—messages are not displayed. |
| **-r** | Maintain a single transaction log for databases that maintain a transaction log mirror. |
| **-t** *log-name* | Specify the file name 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. |
| **-x** n | Reset the transaction log current relative offset to n, so that the database can take part in replication. This option is used for reloading SQL Remote consolidated databases. |

| Option | Description |
|---|---|
| **-z** n | Reset the transaction log starting offset to n, so that the database can take part in replication. This option is used for reloading SQL Remote consolidated databases. |

*Usage*

The **dblog** command line utility allows you to display or change the name of the transaction log or transaction log mirror. You can also stop or start maintaining a transaction log mirror.

The name of the transaction log is set when you create a database. The database cannot be running when you change its transaction log file name.

When you use the **RESTORE** statement to move and/or rename a database, you can rename all of the files except the transaction log. Transactions continue to be written to the old log file name, in the location where the catalog store file (the `.db` file) is located after the database is restored.

When you rename or move all other files in the database, Sybase recommends that you do the same for the log file. Use dblog to move or rename the log file. Run this utility after using **RESTORE** with:

- A new database name
- The **RENAME** option

You can use **dblog** to rename the transaction log even if you have not restored the database, given these restrictions:

- The IQ server must be stopped.
- After the log is renamed, retain the old log until the next database backup, in case the old log is needed for recovery from a media failure.

**dblog** displays additional information about the transaction log:

- Version number
- Starting offset, for use in replication
- Ending offset, for use in replication
- Page size
- Total number of pages
- Number of empty pages
- Percentage of the log file in use

# dbping Database Administration Utility

The **dbping** utility is a ping utility that assists in diagnosing connection problems.

*Syntax*

```
dbping [options]
```

*Parameters*

This table lists the available options for dbping.

**Table 23. dbping options**

| Option | Description |
|---|---|
| **-c** *"keyword=value; ..."* | Supply database connection parameters. See *System Administration Guide: Volume 1 > Connection and Communication Parameters > Connection Parameters*. If no connection parameters are specified, connection parameters from the SQLCONNECT environment variable are used, if set. |
| **-d** | Make a database connection if the server is found.<br><br>If you do not supply the **-d** option, **dbping** reports success if it finds the server specified by the **-c** option. If you do supply the **-d** option, **dbping** reports success only if it connects to both server and database.<br><br>For example, if you have a server named `blair` running the `iqdemo` database, this command succeeds:<br><br>`dbping -c "eng=blair;dbn=iqdemo"`<br><br>The following command fails, with the message `Ping database failed -- specified database not found`<br><br>`dbping -d -c "eng=blair;dbn=iqdemo"` |
| **-en** | Specifies that you want dbping to exit with a failed return code when NULL is returned for any of the properties specified. By default, dbping prints NULL when the value for a property specified by **-pc**, **-pd**, or -ps is unknown, and exits with a success return code. This option can only be used with **-pc**, **-pd**, and -ps. |

| Option | Description |
|---|---|
| **-l** *library* | Use the specified ODBC driver or driver manager library. Specify the library to use (without its file extension). This option does not use the ODBC Driver Manager, and so is particularly useful on UNIX operating systems.<br><br>For example, this command directly loads the ODBC driver:<br>`dbping -m -c "dsn=IQ15iqdemo" -l dbodbc11`<br><br>Use **dbping** to verify connectivity with the ODBC Driver Manager on UNIX systems. Unlike Interactive SQL or other tools, **dbping** allows you to explicitly test the components as you would with a third-party tool. For example:<br>`dbping -m -c "dsn=dsnname" -l /<full path>/ libodbc.so`<br><br>where `libodbc.so` is the third-party ODBC driver. |
| **-m** | Use the ODBC Driver Manager. Otherwise, connect using Embedded SQL™. Establish a connection using ODBC. By default, **dbping** attempts a connection using the embedded SQL interface. |
| **-o** *filename* | Log output messages to a file. |
| **-pc** *property,…* | Upon connection, display the specified connection properties. Supply the properties in a comma-separated list. You must specify enough connection information to establish a database connection.<br><br>For a list of connection properties, see *SQL Anywhere Server - Database Administration > Administering your database > Database administration utilities > Ping utility (dbping).*<br><br>**Note:** This reference points to SQL Anywhere documentation. |
| **-pd** *property,…* | Upon connection, display the specified database properties. Supply the properties in a comma-separated list. You must specify enough connection information to establish a database connection.<br><br>For a list of database properties, see *SQL Anywhere Server - Database Administration > Administering your database > Database administration utilities > Ping utility (dbping).*<br><br>**Note:** This reference points to SQL Anywhere documentation. |

| Option | Description |
|---|---|
| **-ps** *property,…* | Upon connection, display the specified database server properties. Supply the properties in a comma-separated list.<br><br>For a list of database server properties, see *SQL Anywhere Server - Database Administration > Administering your database > Database administration utilities > Ping utility (dbping).*<br><br>**Note:** This reference points to SQL Anywhere documentation. |
| **-q** | Operate quietly—do not print messages. If **dbping** fails, a message always appears. |
| **-s** | Returns information about the performance of the network between the computer running dbping and the computer running the database server. Approximate connection speed, latency, and throughput are displayed. The **-c** option is usually required to specify the connection parameters to connect to a database on the server. You can only use dbping **-s** for embedded SQL connections. This option is ignored if **-m** or **-l** is also specified. By default, dbping -s loops through the requests for at least one second for each statistic it measures. A maximum of 200 connect and disconnect iterations are performed, regardless of the time they take, to avoid consuming too many resources. On slower networks, it can take several seconds to perform the minimum number of iterations for each statistic. The performance statistics are approximate, and are more accurate when both the client and server computers are fairly idle. |
| **-st** time | This option is the same as **-s**, except that it specifies the length of time, in seconds, that dbping loops through the requests for each statistic it measures. This option allows more accurate timing information to be obtained than **-s**. |
| **-z** | Display debugging information. This option is available only when an embedded SQL connection is being attempted. That is, it cannot be combined with **-m** or **-l**. It displays the network communication protocols used to attempt connection, and other diagnostic messages |

### *Usage*

Use **dbping** to help debug connection problems. You can enter a full or partial connection string; the utility returns a message indicating whether the attempt to locate a server or database, or to connect, was successful.

Use **dbping** for Embedded SQL or ODBC connections. You cannot use dbping for jConnect (TDS) connections.

Exit codes are 0 (success) or nonzero (failure).

This utility accepts @filename parameters.

# dbstop Database Shutdown Utility

**dbstop** stops a database server. The dbstop utility is a command-line utility only, available on both UNIX and Windows platforms.

In UNIX, **dbstop** can shut down a server on any node on the network. You must specify a *server-name*, as well as any connection parameters you specified when you started the server. Without the proper connection parameters, **dbstop** does not know how to connect to the server to tell it to shutdown.

The **dbstop** command-line options let you control whether a server is stopped, even if there are active connections.

For full details on when to use **dbstop**, see *System Administration Guide: Volume 1 > Sybase IQ Startup > Database Server Shutdown*.

## Stopping the Database Server with dbstop

Run **dbstop** from the command line.
Issue a command in this format:

```
dbstop [ options ] server-name
```

For example, to stop a database named `iqdemo` on the server **myserver**, enter:

```
dbstop -c "uid=DBA;pwd=sql;eng=myserver;dbn=iqdemo"
```

The following example stops a server, **myserver**, regardless of the database running:

```
dbstop -c "uid=DBA;pwd=sql;eng=myserver;dbn=utility_db"
```

## dbstop Options

This table lists the options available for the **dbstop** utility.

**Table 24. dbstop options**

| Switch | Description |
|---|---|
| @*filename* | Read in options from the specified environment variable or configuration file. |
| server-name | Server name of a running server to stop. If you supply a server name, do not supply connection parameters as well. |

| Switch | Description |
|---|---|
| **-c** *"keyword=value; ..."* | 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; you can use the **-gk** server command-line option to change the default behavior.<br><br>For a description of the connection parameters, see *System Administration Guide: Volume 1 > Connection and Communication Parameters*.<br><br>If there are active connections, **dbstop** prompts whether you want to shut down the server. If you specify `unconditional=true` on the command line, the server shuts down without prompting, even if there are active connections. |
| **-d** | Does not stop the database server. Instead, only stops the database specified in the connection string. |
| **-o** *filename* | Logs output messages to the named file. |
| **-q** | Quiet mode—does not print messages. |
| **-x** | Does not stop if there are active connections. Including this option prevents **dbstop** from prompting for confirmation if there are active connections. |
| **-y** | Stops without prompting even if there are active connections. |

# dbtran Database Administration Utility

Use the **dbtran** log translation utility to translate a transaction log into a `.sql` command file.

You access the Log Translation utility at the command prompt, using the **dbtran** command.

*Syntax*
Running against a database server:

```
dbtran [ options ] -c { connection-string } -n SQL-file
```

Running against a transaction log:

```
dbtran [ options ] [ transaction-log ] [ SQL-file ]
```

*Parameters*

| Option | Description |
|---|---|
| *@data* | Reads in options from the specified environment variable or configuration file. |
| **-a** | Controls whether uncommitted transactions appear in the transaction log. The transaction log contains changes made before the most recent **COMMIT** by any transaction. Changes made after the most recent COMMIT are not present in the transaction log. If you do not specify **-a**, only committed transactions appear in the output file. If you specify **-a**, any uncommitted transactions found in the transaction log are output followed by a **ROLLBACK** statement. |
| **-c** *"keyword=value; ..."* | Specifies the connection string when running the utility against a database server. See *System Administration Guide: Volume 1 > Connection and Communication Parameters > Connection Parameters*. |
| **-d** | Specifies that transactions are written in order from earliest to latest. This feature is provided primarily for use when auditing database activity: do not apply **dbtran** output against a database. |

| Option | Description |
|---|---|
| **-ek** *key* | Specifies the encryption key for strongly encrypted databases. If you have a strongly encrypted database, you must provide the encryption key to use the database or transaction log. Specify either **-ek** or **-ep**, but not both. The command fails if you do not specify the correct encryption key. If you are running dbtran against a database server using the **-c** option, specify the key using a connection parameter instead of using the **-ek** option. For example, the following command gets the transaction log information about database enc.db from the database server sample, and saves its output in `log.sql`.<br><br>**dbtran -n log.sql -c "ENG=sample;DBF=enc.db;UID=DBA;PWD=sql;DBKEY=mykey"** |
| **-ep** | Prompts for the encryption key. This option causes a window to appear, in which you enter the encryption key. It provides an extra measure of security by never allowing the encryption key to be seen in clear text. Specify either -ek or -ep, but not both. The command fails if you do not specify the correct encryption key. If you are running dbtran against a database server using the -c option, specify the key using a connection parameter, instead of using the -ep option. For example, the following command gets the transaction log information about database enc.db from the database server sample, and saves its output in `log.sql`.<br><br>**dbtran -n log.sql -c "ENG=sample;DBF=enc.db;UID=DBA;PWD=sql;DBKEY=mykey"** |
| **-f** | Outputs only transactions that were completed since the last checkpoint. |
| **-g** | Adds auditing information to the transaction log if the auditing database option is turned on. See *Reference: Statements and Options > Database Options > Alphabetical List of Options > AUDITING Option [database].* |
| **-ir** *offset1,offset2* | Outputs a portion of the transaction log between two specified offsets. |

| Option | Description |
|---|---|
| **-is** *source,...* | Outputs operations on rows that have been modified by operations from one or more of the following sources, specified as a comma-separated list:<br>• All All rows. This is the default setting.<br>• SQLRemote Include only rows that were modified using SQL Remote. You can also use the short form “SR”.<br>• RepServer Include only rows that were modified using the Replication Agent (LTM) and Replication Server. You can also use the short form “RS”.<br>• Local Include only rows that are not replicated. |
| **-it** *owner.table,...* | Outputs those operations on the specified, comma-separated list of tables. Specify each table as owner.table. |
| **-j date/time** | Translates only transactions from the most recent checkpoint prior to the given date or time. The user-provided argument can be a date, time, or date and time, enclosed in quotes. you omit a time, the default is the beginning of the day. If you omit a date, the current day is the default. The acceptable format for the date and time is: "YYYY/MMM/DD HH:NN". |
| **-k** | Prevents partial .sql files from being erased if an error is detected. If an error is detected while dbtran is running, the `.sql` file generated until that point is normally erased to ensure that a partial file is not used by accident. Specifying this option may be useful if you are attempting to salvage transactions from a damaged transaction log. |
| **-m** | Specifies a directory that contains transaction logs. Use this option with the **-n** option. |
| **-n** *filename* | Specifies the output file that holds the SQL statements when you run the dbtran utility against a database server. |
| **-o** *filename* | Writes output messages to the named file. |
| **-r** | Removes any transactions that were not committed. This is the default behavior. |
| **-rsu** *username,...* | Specifies a comma-separated list of user names to override the default Replication Server user names. By default, the **-is** option assumes the default Replication Server user names of dbmaint and sa. |

| Option | Description |
|---|---|
| **-s** | Controls how **UPDATE** statements are generated. If you do not use this option, and there is no primary key or unique index on a table, dbtran generates UPDATE statements with a nonstandard FIRST keyword in case of duplicate rows. If you do use this option, the FIRST keyword is omitted for compatibility with the SQL standard. |
| **-sr** | Places generated comments in the output file describing how SQL Remote distributes operations to remote sites. |
| **-t** | Controls whether triggers are included in the command file. By default, actions performed by triggers are not included in the command file. If the matching trigger is in the database, when the command file is run against the database, the trigger performs 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 run. |
| **-u** *userid,...* | Limits the output from the transaction log to include only specified users. |
| **-x** *userid,...* | Limits the output from the transaction log to exclude specified users. |
| **-y** | Replaces existing command files without prompting you for confirmation. If you specify **-q**, you must also specify **-y** or the operation fails. |
| *transaction-log* | Specifies the log file to be translated. Cannot be used together with **-c** or **-m** options |
| *SQL-file* | Names the output file containing the translated information. For use with *transaction-log* only. |

*Usage*

**dbtran** 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—When you run dbtran against a database server, 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. DBA authority is required to run in this way. For example, this command translates log information from the server **iqdemo** and places the output in a file named `iqdemo.sql`:

```
dbtran -c "eng=iqdemo;dbn=iqdemo;dbf=iqdemo.db;uid=DBA;pwd=sql" -n
iqdemo.sql
```

- Against a transaction log file—When you run dbtran against a transaction log, the utility acts directly against a transaction log file. Protect your transaction log file from general access to prevent users from running this statement.

  ```
  dbtran iqdemo.log iqdemo.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.

If **-c** is used, dbtran attempts to translate the online transaction log file, and all the offline transaction log files in the same directory as the online transaction log file. If the directory contains transaction log files for more than one database, dbtran may give an error. To avoid this, ensure that each directory contains transaction log files for only one database.

A transaction can span multiple transaction logs. If transaction log files contain transactions that span logs, translating a single transaction log file (for example, dbtran demo.log) can cause the spanning transactions to be lost. For dbtran to generate complete transactions, use the -c or **-m** options with the transaction log files in the directory.

Exit codes are 0 (success) or nonzero (failure).

This utility accepts @filename parameters.

# dbvalid Database Administration Utility

The dbvalid utility is a validation utility that validates the indexes and keys on some or all of the SQL Anywhere tables in the catalog store.

The Validation utility scans the entire table and looks up each record in every index and key defined on the table. By default, the Validation utility uses the express check option.

**Note:** The **dbvalid** utility lets you easily validate SQL Anywhere catalog store tables, but does not validate IQ tables. Use the IQ stored procedure **sp_iqcheckdb** to validate IQ tables.

You can access the **dbvalid** utility at the system command-line level, which is useful for incorporating **dbvalid** into batch or command files.

### *Syntax*

```
dbvalid [ options ] [object-name,... ]
```

### *Parameters*

This table lists the options available for the **dbvalid** utility.

**Table 25. dbvalid options**

| Option | Description |
|---|---|
| *object-name* | The name of a table or (if **-i** is used) an index to validate |
| **-c** "*keyword=value*; ..." | Supply database connection parameters. See *System Administration Guide: Volume 1 > Connection and Communication Parameters > Connection and Communication Parameters*. The user ID must have `DBA` authority or REMOTE DBA authority.<br><br>For example, the following validates the iqdemo database, connecting as user `DBA` with password `sql`:<br><br>`dbvalid -c “uid=DBA;pwd=sql;dbf-c:\sybase \IQ-15_4\demo\iqdemo.db”` |
| **-o** *filename* | Log output messages to a file |
| **-f** | Validate tables with full check. In addition to the default validation checks, carry out both data checks (**-fd**) and index checks (**-fi**). This option corresponds to the **WITH FULL CHECK** option of the SQL Anywhere **VALIDATE TABLE** statement. Depending on the contents of your catalog store, this option may significantly extend the time required to validate. |

| Option | Description |
|---|---|
| **-fd** | Validate tables with data check. In addition to the default validation checks, check that all of each `LONG BINARY`, `LONG VARCHAR`, `TEXT` or `IMAGE` data type can be read. Entries with these data types may span more than one page. In the IQ catalog store:<br>• Domain – user-defined data type.<br>• `IMAGE` – a domain to `LONG BINARY.`<br>• `TEXT` – a domain to `LONG VARCHAR.`<br>This option instructs the database server to check all pages used by each entry. This corresponds to the WITH DATA CHECK option on the SQL Anywhere **VALIDATE TABLE** statement. Depending on the contents of your catalog store, this option may significantly extend the time required to validate. |
| **-fi** | Validate tables with index check. In addition to the default validation checks, validate each index on the table. This corresponds to the **WITH INDEX CHECK** option of the SQL Anywhere **VALIDATE TABLE** statement. Depending on the contents of your catalog store, this option may significantly extend the time required to validate. |
| **-fx** | Validate tables with express check. In addition to the default and data checks, check that the number of rows in the table matches the number of entries in the index. This corresponds to the WITH EXPRESS CHECK of the SQL Anywhere **VALIDATE TABLE** statement. This option does not perform individual index lookups for each row. |
| **-i** | Each *object-name* is an index. Instead of validating tables, validate indexes. Ensure that every row referenced in the index actually exists in the table. For foreign-key indexes, **-i** also ensures that the corresponding row exists in the primary table. If you supply a *table-name* instead of an *index-name,* validates the primary key index. In this case, for **dbvalid**, each of the *object-name* values supplied represents an index instead of a table and has a name in the following format:<br>`[ [ owner.]table-name.]index-name`<br>Must be the owner of the table on which the index is created, have `DBA` authority, or have REMOTE DBA authority. |
| **-q** | Operate quietly—do not print output messages. |

| Option | Description |
|---|---|
| **-s** | Validate database pages using checksums. Checksums are used to determine whether a database page has been modified on disk. If you created a database with checksums enabled, you can validate the catalog store using checksums. Checksum validation reads each page of the catalog store from disk and calculates its checksum. If the calculated checksum is different from the checksum stored on the page, the page has been modified on disk and an error is returned. The page numbers of any invalid catalog store pages appear in the server messages window. You cannot use **-s** with **-i**, **-t**, or any of the **-f** options. |
| **-t** | The list of *object-name* values is a list of tables. This is also the default behavior. |

*Usage*

With the **dbvalid** command-line utility, you can validate the indexes and keys on some or all of the SQL Anywhere tables in the catalog store. dbvalid scans the entire table and confirms that each row exists in the appropriate indexes. It is the same as running the SQL Anywhere **VALIDATE TABLE** statement on each catalog store table.

**Note:** **VALIDATE TABLE** is not supported in Sybase IQ. **sp_iqcheckdb** provides a similar function for IQ store tables.

By default, the Validation utility uses the express check option. However, the express check option is *not* used if you specify **-f**, **-fd**, **-fi**, **-fn**, or **-i**.

If the catalog store table is inconsistent, **dbvalid** reports an error. If errors are reported, you can drop all of the indexes and keys on a table and re-create them. You must also re-create any foreign keys to the table.

**Warning!** Validate a table or entire catalog store only when no connections are making changes to the database; otherwise, spurious errors may be reported, indicating some form of database inconsistency even though no inconsistency actually exists.

| Program exit code | Description |
|---|---|
| 0 | Database validated successfully |
| 1 | General failure in utility |
| 2 | Error validating database |
| 7 | Cannot find database to connect to (database name is wrong) |
| 8 | Cannot connect to database (user ID/password is wrong) |
| 11 | Cannot find server to connect to (server name is wrong) |

| Program exit code | Description |
|---|---|
| 12 | Incorrect encryption key for starting database |

*Example*

The following command validates the catalog store of the iqdemo database, connecting as user `DBA` with password `sql`:

```
dbvalid -c “uid=DBA;pwd=sql;dbf-c:\sybase\IQ-15_4\demo\iqdemo.db”
```

# iqdscp Configuration Utility

The **iqdscp** utility is a UNIX-only Open Client and Open Server configuration utility you can use to configure the interfaces file and to configure a directory service.

See *Open Server 15.5 > Open Client and Open Server Configuration Guide for UNIX > Using dscp*.

# iqdsedit Database Administration Utility

The **iqdsedit** utility allows you to configure the interfaces file (interfaces or SQL.ini).

See *SQL Anywhere Server - Database Administration > Replication > Using SQL Anywhere as an Open Server > Configuring Open Servers > Using the DSEdit utility*.

**Note:** This reference points to SQL Anywhere documentation.

# iqdsn Database Administration Utility

The **iqdsn** utility is a data source utility you use for creating, changing, deleting, describing, and listing Sybase IQ ODBC data sources.

The **iqdsn** utility is a cross-platform alternative to the ODBC Administrator.

On Windows operating systems, data sources are held in the Registry. On UNIX operating systems, data sources are held in the `.odbc.ini` file. Use the **iqdsn** utility for batch operations.

### *Syntax*

```
iqdsn [ modifier-options ]
{ -l
| -d dsn
| -g dsn
| -w dsn [details-options;...]
| -cl }
```

### *Parameters*

These tables list the available options for the **iqdsn** utility.

**Table 26. iqdsn major options**

| Major option | Description |
|---|---|
| **-l** | Lists the available Sybase IQ ODBC data sources. You can modify the list format using the **-b** or **-v** options. |
| **-d** *dsn* | Deletes the named data source. If you supply **-y**, any existing data source is overwritten without confirmation. |
| **-g** *dsn* | List the definition of the named data source. You can modify the format of the output using the **-b** or **-v** option. |
| **-w** *dsn* [ *details-options* ] | Creates a new data source, or overwrites one if one of the same name exists. If you supply **-y**, any existing data source is overwritten without confirmation. |
| **-cl** | This convenience option lists the connection parameters supported by the **iqdsn** utility. |

**Table 27. iqdsn modifier-options**

| Modifier option | Description |
|---|---|
| **-b** | Brief. Format the output of the list as a single line connection string. |

| Modifier option | Description |
|---|---|
| **-or** | Creates a data source for the iAnywhere Solutions Oracle driver when specified with the **-c** option.<br>For example:<br>`dbdsn -w MyOracleDSN -or -c Userid=DBA;Pass- word=sql;SID=abcd;ArraySize=500;ProcResults=y`<br>You can specify the **-cl** option with the **-or** option to obtain a list of the connection parameters for the iAnywhere Solutions Oracle driver.<br>See *MobiLink - Server Administration > MobiLink Reference > iAnywhere Solutions ODBC drivers for MobiLink > iAnywhere Solutions 12 - Oracle ODBC driver*. |
| **-q** | Quiet. Do not print the informational banner. |
| **-v** | Verbose. Format the output of the list over several lines, as a table. |
| **-va** | Verbose All. Print connection parameters in same format as **-v**, but also include other hidden parameters. Use this option to display ODBC driver qualifier needed for remote data access on those UNIX platforms that support such access, or for some third-party driver managers. |
| **-y** | Automatically delete or overwrite each file without prompting you for confirmation. |

**Table 28. iqdsn details-options**

| Details option | Description |
|---|---|
| **-c "keyword=***value*;...**"** | Specify connection parameters as a connection string. See *System Administration Guide: Volume 1 > Connection and Communication Parameters*. |
| **-ec** *encryption type* | Encrypt packets sent between the client application and the server. See *System Administration Guide: Volume 1 > Connection and Communication Parameters > Connection Parameters > Encryption Connection Parameter [ENC]*. |
| **-o** *filename* | Write output messages to the named file. By default, messages are written to the console. See *System Administration Guide: Volume 1 > Connection and Communication Parameters > Connection Parameters > LogFile Connection Parameter [LOG]*. |

| Details option | Description |
|---|---|
| **-p** *size* | Set the maximum packet size for network communications, in bytes. The value must be greater than 500, and less than 16000. The default setting is 1460. See *System Administration Guide: Volume 1 > Connection and Communication Parameters > Connection Parameters > CommBuffer-Size Connection Parameter [CBSize].* |
| **-r** | Disable multiple record fetching. By default, when the database server gets a simple fetch request, the application asks for extra rows. Use the **-r** option to disable this behavior. See *System Administration Guide: Vol-ume 1 > Connection and Communication Parameters > Connection pa-rameters > DisableMultiRowFetch Connection Parameter [DMRF].* |
| **-tl** *seconds* | Client liveness timeout period. Terminates connections when they are no longer intact. The value is in seconds. The default is the server setting, which in turn has a default of 120 seconds. See *System Administration Guide: Volume 1 > Connection and Communication Parameters > Con-nection Parameters > LivenessTimeout Connection Parameter [LTO].* |
| **-x** *list* | List network drivers to run. |
| **-z** | Provide diagnostic information on communications links on startup. |
| *server-name* | Connect to the named database server. Only the first 40 characters are used. |

### Usage

The **iqdsn** modifier options can occur before or after the major option specification. The order makes a difference only if you specify a a connection parameter value more than once. In such a case, the last value specified is used.

### Examples

This example writes a definition of the data source `newdsn` and does not prompt for confirmation if the data source already exists.

```
iqdsn -y tcpip -w newdsn -c "uid=DBA;pwd=sql" -v
```

You can also change the order of options:

```
iqdsn -w newdsn -c "uid=DBA;pwd=sql" -x tcpip -y
```

Lists all known user data sources, one data source name per line:

```
iqdsn -l
```

Lists all data sources along with their associated connection string:

```
iqdsn -l -b
```

Reports the connection string for user data source `MyDSN`:

```
iqdsn -g MyDSN
```

Deletes the data source `BadDSN`, but first list the connection parameters for `BadDSN` and prompt for confirmation:

```
iqdsn -d BadDSN -v
```

Deletes the data source `BadDSN` without prompting for confirmation.

```
iqdsn -d BadDSN -y
```

Creates a data source named `NewDSN` for the database server `MyServer`:

```
iqdsn -w NewDSN -c "uid=DBA;pwd=sql;eng=bar"
```

If a `NewDSN` already exists, the utility asks you if you want to overwrite it.

The following example connects to the `sample` database server. The server name **sample** overrides the previous specified value of `MyServer`:

```
iqdsn -w NewDSN -c "uid=DBA;pwd=sql;eng=MyServer" sample
```

Lists all connection parameter names and their aliases:

```
iqdsn -cl
```

# iqheader Database Administration Utility

The **iqheader** utility is a dbspace header utility that determines which server, if any, is using a particular device, file, or LUN (Logical Unit Number) as a dbspace to analyze disk usage or to configure a multiplex query server.

The **iqheader** utility reports the configuration of an arbitrary device regardless of whether it is currently in use by an IQ server.

The user interface is a standalone console application called **iqheader** (**header.exe** on Windows). The iqheader tool searches the device for an IQ dbspace header and reports the header information in a user-readable format.

**Note:** LUN is a logical unit number and is used to identify SCSI devices so the host can address and access the data on each disk drive in an array.

### *Syntax*

**iqheader** `[ [` *`dbspace_path`* `]`

### *Parameters*

The **iqheader** application takes a single parameter, which is the device to be checked.

### *Usage*

- **iqheader usage –** When invoked with no parameters, a usage summary is reported and a nonzero status is returned:

  ```
  >iqheader
  Usage: iqheader [dbspace_path]
  ```

- **iqheader error –** When the specified target is not an IQ dbspace, an error message is reported and a nonzero status is returned:

  ```
  >iqheader /dev/null
  Not an IQ file: Error 0
  ```

- **Operating system errors –** When the specified target is unreadable, or any file operation fails due to an error returned from the operating system, the native operating system error appears and a nonzero status is returned:

  ```
  >iqheader /dev/rdsk/c1t32d0s1              <
  Open Failed: No such file or directory

  >iqheader /dev/rdsk/c1t3d0s1               <
  Open Failed: Permission denied
  ```

- **iqheader output –** When a valid IQ dbspace is specified, **iqheader** prints the dbspace configuration to the console and returns a 0 exit status.

**Table 29. iqheader output**

| Field | Description |
|---|---|
| File Name | Name of the file. |
| Full Path | Full path after symbolic link resolution |
| Version | Dbspace file format version |
| File ID | Unique number assigned to each dbspace |
| Create Time | Time of dbspace creation |
| RW Mode | Current read-write mode: RW, RO, RW, N/A (Upgraded) |
| Last RW Mode | Last dbspace mode |
| Size (MB) | DBSpace size, in megabytes |
| Reserve (MB) | DBSpace reserve size, in megabytes |
| Block Size | Size of IQ block, in bytes |
| Page Size | Size of IQ page, in bytes |
| First Block | First IQ block number mapped to dbspace |
| Block Count | Number of IQ blocks that map to actual disk blocks |
| Reserve Blocks | Number of IQ blocks that may be added to this dbspace |
| Last Real Block | Last IQ block number that maps to an actual disk block |
| Last Mapped Block | Last IQ block number mapped to dbspace |
| OFlag | Online status (YES/NO) |
| Create ID | Commit ID in which dbspace was created |
| Alter ID | Last commit ID in which dbspace was altered |
| DBID1 | Location of first database identity |
| DBID2 | Location of second database identity |
| DBSpace ID | Unique identifier assigned to each dbspace |
| _NextFLAllocLowerBank | Lower bound of pre-allocate space for dbspace |
| _NextFLAllocUpperBank | Upper bound of pre-allocate space for dbspace |
| Pre-alter commit ID | Commit_id in the system tables ISYSDBFILE and ISYSIQDBSPACE |
| _ReqNumFreeListBlocks | Number of blocks of type 'F' (free list blocks) |

*Example*

This example shows output for iqheader:

```
File Name: file1.iq
Full Path: /dev/dsk/file1.iq
DBFile Header Info
Version: 2
File ID: 16395
Create Time: 2008-06-02 21:57:00
RW Mode: RO
Last RW Mode: RW
Size (MB): 20
Reserve (MB): 20
Block Size: 8192
Page Size: 131072
First Block: 9408960
Block Count: 2560
Reserve Blocks: 2560
Last Real Block: 9411519
Last Mapped Block: 10454399
OFlags: 1
Create ID: 6905
Alter ID: 6964
DBID1: 0
DBID2: 0
DBSpace ID: 16395
_NextFLAllocLowerBank: 0
_NextFLAllocUpperBank: 0
Pre-alter commit ID: 6925
Dropped: NO
```

# iqinit Database Administration Utility

The **iqinit** utility starts a database that gives the user executing the utility permission to create a database. The user must have access to the computer and file system.

**iqinit** is the IQ version of the SQL Anywhere **dbinit** utility. **iqinit** lets you create either an IQ or SQL Anywhere database from the command line without starting a database:

- If no **iqinit** command line parameters are specified, **iqinit** creates a SQL Anywhere database.
- If the **-iqpath** command line parameter is specified, **iqinit** creates an IQ database.

### *Syntax*

```
iqinit  [ options ] new-database-file
```

### *Parameters*

This table lists the available options for the **iqinit** utility.

**Table 30. iqinit options**

| Option | Description |
|---|---|
| *@data* | For descriptions of options common to **dbinit** and **iqinit**, see *SQL Anywhere Server - Database Administration > Administering Your Database > Database administration utilities > Initialization utility (dbinit)*. |
| **-a** | |
| **-af** | |
| **-b** | |
| **-c** | |
| **-dba** [ *DBA-user* ] [ *pwd* ] | |
| **-dbs** *size*[ **k** \| **m** \| **g** \| **p** ] | |
| **-ea** *algorithm* | |
| **-ek** *key* | |
| **-ep** | |
| **-et** | |
| **-i** | |
| **-k** | |
| **-l** | |

| Option | Description |
|---|---|
| **-le** | |
| **-m** *filename* | |
| **-n** | |
| **-o** *filename* | |
| **-p** *page-size* | |
| **-q** | |
| **-s** | |
| **-t** *transaction-log-name* | |
| **-z** *coll* [ *collation-tailoring-string* | |
| **-ze** *encoding* | |
| **-zn** *coll* [ *collation-tailoring-string* | |
| **-iqblksize** | The I/O transfer block size in bytes. |
| **-iqmsgpath** | The path name of the segment containing the Sybase IQ message trace file. |
| **-iqpath** | The path name of the main segment file containing the IQ data. |
| **-iqpgsize** | The page size in bytes for the Sybase IQ segment of the database. |
| **-iqreservesize** | The size in MB of the space to reserve for the IQ main store. |
| **-iqsize** | The size in MB of either the raw partition or OS file with the **-iq-path**. |
| **-iqtmppath** | The path name of the temporary segment file. |
| **-iqtmpreservesize** | The size in MB to reserve for the temporary IQ store. |
| **-iqtmpsize** | The size in MB of either the raw partition or OS file for the **-iqtmp-path**. |

*Example*

This command creates a Sybase IQ database called `bar.iq`.

```
$ iqinit -iqpath bar.iq -iqsize 20M
-iqpgsize 2048
-iqreservesize 10M bar.db
```

```
SQL Anywhere Initialization Utility Version 15.3.0.5530
Debug
CHAR collation sequence:
ISO_BINENG(CaseSensitivity=Respect)
CHAR character set encoding:  ISO_8859-1:1987
NCHAR collation sequence:
UCA(CaseSensitivity=UpperFirst;
AccentSensitivity=Respect;
PunctuationSensitivity=Primary)
NCHAR character set encoding:  UTF-8
Creating system tables
Creating system views
Setting option values
Database "bar.db" created successfully
```

# iqisql Interactive SQL Utility

**iqisql** and **isql** are the same utility with different names. **iqisql** will be retired in a future version of Sybase IQ. Sybase recommends that you use the **isql** utility instead.

Behavior differences may exist between **iqisql** and **isql** if they originate in different versions of Sybase IQ. The two utilities are functionally equivalent if you access them from the same Sybase IQ version.

### See also

- *isql Interactive SQL Utility* on page 113
- *dbisql Interactive SQL Utility* on page 13
- *Appendix: dbisqlc Interactive SQL Classic Utility (Deprecated)* on page 197

# iqocscfg Configuration Utility

The **iqocscfg** utility is a Windows-only Open Client and Open Server configuration utility you can use to configure environment variables, directory drivers, and security drivers.

See *Open Server 15.5 > Software Developer's Kit 15.5 > Open Client and Open Server Configuration Guide for Microsoft Windows > Using ocscfg*.

# iqsqlpp SQL Preprocessor Utility

The Sybase IQ SQL preprocessor utility **iqsqlpp** translates the SQL statements in an input file (`.sqc`) into C language source that is put into an output file (`.c`).

Embedded SQL is a database programming interface for the C and C++ programming languages. Embedded SQL consists of SQL statements intermixed with (embedded in) C or C++ source code. These SQL statements are translated by an SQL preprocessor into C or C++ source code, which you then compile.

*Syntax*

```
iqsqlpp [ options ] <in filename > [<out filename> ]
```

*Parameters*

This table lists the options available for the **iqsqlpp** utility.

**Table 31. iqsqlpp options**

| Option | Description |
|---|---|
| **-d** | Favor data size. |
| **-e** *<level>* | Flag nonconforming SQL syntax as an error.<br><br>The allowed values of <level> are:<br><br>• **c03**– Flag syntax that is not core SQL/2003 syntax<br>• **p03** – Flag syntax that is not full SQL/2003 syntax<br>• **c99** – Flag syntax that is not core SQL/1999 syntax<br>• **p99** – Flag syntax that is not full SQL/1999 syntax<br>• **e92** – Flag syntax that is not entry-level SQL/1992 syntax<br>• **i92** – Flag syntax that is not intermediate-level SQL/1992 syntax<br>• **f92** – Flag syntax that is not full-SQL/1992 syntax<br>• **t** – Flag syntax that is not full-SQL/1992 syntax<br>• **u** – Flag non-standard host variable types<br><br>The following are also supported for compatibility with previous versions: **e** (for entry-level SQL92), **i**, (for intermediate-level SQL92), **f** (for full SQL92), and **w** (to allow all supported syntax). |
| **-h** *<width>* | Limit the maximum line length of output. |
| **-k** | Include user declaration of SQLCODE. |
| **-n** | Line numbers. |

| Option | Description |
|---|---|
| **-o** *<O/S spec>* | Target operating system specification (WINDOWS, WINNT or UNIX). |
| **-q** | Quiet mode—do not print banner. |
| **-r-** | Generate reentrant code. |
| **-s** *<len>* | Maximum string constant length for the compiler. |
| **-w** <level> | Flag nonconforming SQL syntax as a warning.<br>The allowed values of <level> are:<br>• **c03**– Flag syntax that is not core SQL/2003 syntax<br>• **p03** – Flag syntax that is not full SQL/2003 syntax<br>• **c99** – Flag syntax that is not core SQL/1999 syntax<br>• **p99** – Flag syntax that is not full SQL/1999 syntax<br>• **e92** – Flag syntax that is not entry-level SQL/1992 syntax<br>• **i92** – Flag syntax that is not intermediate-level SQL/1992 syntax<br>• **f92** – Flag syntax that is not full-SQL/1992 syntax<br>• **t** – Flag syntax that is not full-SQL/1992 syntax<br>• **u** – Flag non-standard host variable types |
| **-x** | Change multibyte SQL strings to escape sequences. |
| **-z** *<cs>* | Specify the collation sequence. For a list of recommended collation sequences, enter **iqinit -l** at a command prompt. |

### *Usage*

The SQL preprocessor processes a C or C++ program containing Embedded SQL before the compiler is run. **iqsqlpp** translates the SQL statements in the input file *sql-filename* into C language source that is put into the *output-filename*. The normal extension for source programs with Embedded SQL is `.sqc`. The default output file name is the *sql-filename* with an extension of `.c`. If the *sql-filename* has a `.c` extension, the default output file name extension is `.CC`.

### *Options*

- **-d –** Favor data size. Generate code that reduces data space size. Data structures are reused and initialized at execution time before use. This increases code size.
- **e <flag>- –** This option flags, as an error, any Embedded SQL that is not part of a specified set of SQL92.

  The allowed values of *<flag>* and their meanings are as follows:

  - **e –** Flag syntax that is not entry-level SQL92 syntax.

  - **i –** Flag syntax that is not intermediate-level SQL92 syntax.
  - **f –** Flag syntax that is not full-SQL92 syntax.
  - **t –** Flag non-standard host variable types.
  - **w –** Allow all supported syntax.
  - **u –** Flag syntax that is not supported by UltraLite.
  - **c99 –** Flag syntax that is not core SQL/1999 syntax.
  - **c03 –** Flag syntax that is not core SQL/2003 syntax.
  - **p99 –** Flag syntax that is not full SQL/1999 syntax.
  - **p03 –** Flag syntax that is not full SQL/2003 syntax.

- **-h width –** Limits the maximum length of lines output by **iqsqlpp** to *width*. The continuation character is a backslash (\), and the minimum value of *width* is ten.
- **-k –** Notifies the preprocessor that the program to be compiled includes a user declaration of SQLCODE.
- **-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 you are using supports the *#line* directive, this option makes the compiler report errors on line numbers in the SQC file (the file 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 are used indirectly by the source level debugger so that you can debug while viewing the SQC source file.
- **-o –** <O/S spec> Specify the target operating system. This option must match the operating system where you run the program. A reference to a special symbol is 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 is detected by the linker. The supported operating systems are:
  - **WINDOWS –** Microsoft Windows
  - **UNIX –** Use this option if you are creating a 32-bit UNIX application.
  - **UNIX64 –** Use this option if you are creating a 64-bit UNIX application.
- **-q –** Operate quietly. Do not print the banner.
- **-r- –** Generate reentrant code. For more information on reentrant code, see *SQL Anywhere Server - Programming > SQL Anywhere Data Access APIs > SQL Anywhere embedded SQL > The SQL Communication Area (SQLCA) > SQLCA management for multi-threaded or reentrant code*.

  **Note:** This reference points to SQL Anywhere documentation.

- **-s <len> –** Set the maximum size string that the preprocessor puts into the C file. Strings longer than this value are initialized using a list of characters (*"a," "b," "c,"* and so on). 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 <flag> –** This option flags any Embedded SQL that is not part of a specified set of SQL92 as a warning.

  The allowed values of *<flag>* and their meanings are as follows:

  - **e –** Flag syntax that is not entry-level SQL92 syntax.
  - **i –** Flag syntax that is not intermediate-level SQL92 syntax.
  - **f –** Flag syntax that is not full-SQL92 syntax.
  - **t –** Flag non-standard host variable types.
  - **u –** Flag syntax that is not supported by UltraLite.
  - **w –** Allow all supported syntax.
  - **c03 –** Flag syntax that is not core SQL/2003 syntax.
  - **c99 –** Flag syntax that is not core SQL/1999 syntax.
  - **p99 –** Flag syntax that is not full SQL/1999 syntax.
  - **p03 –** Flag syntax that is not full SQL/2003 syntax.
- **-x –** Change multibyte strings to escape sequences so that they can pass through compilers.
- **-z <cs> –** This option specifies the collation sequence. For a list of recommended collation sequences, enter **iqinit -l** at a command prompt.

  The collation sequence helps the preprocessor understand the characters used in the source code of the program, for example, in identifying alphabetic characters suitable for use in identifiers. If you do not specify **-z**, the preprocessor attempts to determine a reasonable collation to use, based on the operating system and the IQLANG and IQCHARSET environment variables.

# isql Interactive SQL Utility

**isql** and **iqisql** are the same utility with different names. The **isql** utility is a command-line Interactive SQL utility that uses the Adaptive Server Enterprise Open Client API.

For syntax and parameters, see the Adaptive Server Enterprise documentation: *Adaptive Server Enterprise 15.5 > Utility Guide > Utility Commands Reference > isql*.

Neither **isql** nor **iqisql** let you create user-defined database options. If you need to add your own database options, use the **dbisql** Interactive SQL utility instead.

### See also

- *iqisql Interactive SQL Utility* on page 105
- *dbisql Interactive SQL Utility* on page 13
- *Appendix: dbisqlc Interactive SQL Classic Utility (Deprecated)* on page 197

# start_iq Database Server Startup Utility

The database startup utility **start_iq** starts a Sybase IQ network database server.

The UNIX versions of Sybase IQ provide the script **start_iq**, and the Windows version of IQ provides **start_iq.exe**. These scripts verify that your environment is set correctly and start the server with all required switches preset to recommended defaults (along with any switches you add). The **start_iq** utility also includes some parameters and calculates others. For switches that are specific to your operating system, see the *Installation and Configuration Guide*.

**Note:** The Start Database Server utility in Sybase Central provides an easy graphical interface for starting servers and is the recommended method for starting IQ multiplex servers. See *System Administration Guide: Volume 1 > Sybase IQ Startup > Ways to Start Database Servers*.

## Starting the start_iq Utility

Start **start_iq** from the command line on Windows and UNIX.

1. Issue a command in this format:

```
start_iq [ server-options ] [ database-file
[ database-options ], ...]
```

2. Specify the **-n** switch in [ *server-options* ] to prevent you from unintentionally connecting to the wrong server.

## Listing all start_iq Switches

You can display a list of all available switches for the *server-options*, *database-file*, and *database-options* parameters.
Issue the following command:

```
start_iq -?
```

## start_iq Server Options

Available switches for the **start_iq** *server-options* parameters.

### See also

• *Initial Catalog Store Cache Size when start_iq Server Option -c is not Specified* on page 117

- *Database Server Naming Restrictions* on page 157
- *AWE Cache Allocation* on page 128
- *Starting a Database Server with an AWE Cache* on page 129

## @filename iqsrv15 Server Option

Read in switches from configuration file.

*Syntax*
@*filename*

*Remarks*
The file may contain line breaks, and may contain any set of command-line switches. For example, the following Windows command file holds a set of command-line switches for a server named `iqdemo` that allows 10 connections, and sets the maximum catalog page size to 4096 bytes, and starts the iqdemo database:

```
# iqdemo.cfg
# ------------------------------------------------
# Default startup parameters for the IQ demo database
# ---------------------------------------------------
-n  iqdemo
-x  tcpip{port=2638}
# The following parameters are also found in the configuration file
# %IQDIR%\scripts\default.cfg.  Any parameters not specified below
# and not in the start up parameter list, will be added by start_iq
# using default.cfg as a guide.

-c  48m
-gc 20
-gd all
-gl all
-gm 10
-gp 4096
-iqmc 32
-iqtc 24
```

If this configuration file is saved as `c:\config.txt`, the file can be used in a command line as follows:

```
start_iq @c:\config.txt
```

## @envvar iqsrv15 Server Option

Reads in command-line switches from the supplied environment variable.

*Syntax*
**@envvar**

*Remarks*

The environment variable may contain any set of command-line switches. For example, the first of the following pair of Windows statements sets an environment variable holding a set of command-line switches and loads the sample database. The second statement starts the database server:

```
set envvar= -gp 4096 -gm 15
c:\sybase\IQ-15_4\demo\start_iq -n myserver @envvar iqdemo.db
```

**Note:** If you have both a file and an environment variable with the value of your @ command-line switch, the result is unpredictable. Use only one of these methods to set a given @ command-line switch.

## -c iqsrv15 Server Option

Sets initial memory reserved for caching catalog store pages and other server information.

*Syntax*

**-c** *cache-size[ k| m| g| p]*

*Remarks*

The database server uses extra memory for caching database pages if the memory is set aside in the cache. Any cache size less than 10000 is assumed to be KB (1K = 1024 bytes). Any cache size 10000 or greater is assumed to be in bytes. You can also specify the cache size `nK,` `nM` or `nP` (1M = 1024 KB), where **P** is a percentage of the physical system memory.

The default value of **-c** in the `default.cfg` file and **start_iq** is 32MB (**-c 32M**) for Windows platforms, and 48MB (**-c 48M**) for UNIX platforms. For IQ databases, Sybase recommends that you use this default or set **-c** to a higher value.

You can use % as an alternative to P, but as most non-UNIX operating systems use % as an environment variable escape character, you must escape the % character. For example, to use 20 percent of the physical system memory, specify:

```
start_iq -c 20%% ...
```

Do not use **-c** in the same configuration file or command line with **-ch** or **-cl**. For related information, see the **-ch cache-size** option and the -ca 0 option.

### Initial Catalog Store Cache Size when start_iq Server Option -c is not Specified

If you do not provide a value for the **start_iq** server option **-c** switch (either on the command line or using the **start_iq** default), the database server computes the initial catalog store cache allocation as follows:

- The database server uses 32MB as the minimum default cache size.
- The database server computes a runtime-specific minimum default cache size, which is the lesser of the following items:

- 25% of the physical memory of the machine.
- The sum of the sizes of the main database files specified on the command line. Additional dbspaces apart from the main database files are not included in the calculation. If no files are specified, this value is zero.

- The database server allocates the greater of the two values computed.

**See also**

- *start_iq Server Options* on page 115
- *AWE Cache Sizes per Operating System* on page 127
- *Starting a Database Server with an AWE Cache* on page 129

## -ca iqsrv15 server option

Enforces a static cache size.

*Syntax*

```
iqsrv15 -ca 0 ...
```

*Applies to*

Windows, Unix

*Remarks*

You can disable automatic cache size tuning by specifying -ca 0 option. If you do not include the -ca 0 option, the database server automatically increases the cache size. If you specify this option, the cache size is still adjusted if the database server would otherwise run into an error indicating that the dynamic memory is exhausted.

This server option must only be used in the form -ca 0.

This option is ignored if you are using an AWE cache.

**Note:** The use of AWE is deprecated. It is recommended that you use the 64-bit version of the database server on a 64-bit Windows operating system if you require a large cache.

*Example*

```
start_iq @iqdemo.cfg iqdemo.db
```

The following example is for Windows:

```
start_iq @"%
      ALLUSERSPROFILE%\SybaseIQ\demo\iqdemo.cfg"   -c 40P
      -ca
    0 -n myserver "%
      ALLUSERSPROFILE%\SybaseIQ\demo\iqdemo.db"
```

The following example is for Unix:

```
start_iq
      @$IQDIR15/demo/iqdemo.cfg -c 40P -ca 0
    $IQDIR15/demo/iqdemo.db
```

## -cc iqsrv15 server option

Collects information about database pages to be used for cache warming the next time the database is started.

*Syntax*

```
iqsrv15 -cc{ + | - } ...
```

*Applies to*

All operating systems and database servers.

*Remarks*

By default, page collection is turned on. When collection is turned on, the database server keeps track of each database page that is requested. Collection stops when the maximum number of pages has been collected, the database is shut down, or the collection rate falls below the minimum value. You cannot configure the maximum number of pages collected or specify the value for the collection rate (the value is based on cache size and database size). Once collection stops, information about the requested pages is recorded in the database so those pages can be used to warm the cache the next time the database is started with the -cr option. Collection of referenced pages is turned on by default.

## -ch iqsrv15 Server Option

Set catalog store cache size upper limit in bytes.

*Syntax*

**-ch** *size[ k| m| g| p]*

*Remarks*

By default, the upper limit is approximately the lower of 256MB and 90% of the physical memory of the machine.

You specify the cache-size using the K, M, and P characters as in the **-c** option. For the meaning and usage of the cache size argument and the K, M, and P characters, see *-c cache-size*.

In some cases the standard catalog cache size may be too small, for example, to accommodate certain queries that need a lot of parsing. In these cases, you may find it helpful to set **-cl** and **-ch**. For example, on 32-bit platforms, try these settings:

```
-cl 128M
-ch 512M
```

**Warning!** To control catalog store cache size explicitly, you must do either of the following, but not both, in your configuration file (`.cfg`) or on the UNIX command line for server startup:

- Set the **-c** parameter.
- Set specific upper and lower limits for the catalog store cache size using the **-ch** and **-cl** parameters.

Specifying different combinations of the parameters above can produce unexpected results.

## -cl iqsrv15 server option

Sets a minimum cache size as a lower limit to dynamic cache resizing.

*Syntax*

```
iqsrv15 -cl { size[ k | m | g | p ] } ...
```

*Default*

2 MB on Windows

8 MB on Unix

*Applies to*

Windows, Unix

*Remarks*

This option sets a lower limit to the cache. If -c is specified, and -cl is not specified, then the minimum cache size is set to the initial cache size (the -c setting). If neither -c nor -cl is set, the minimum cache is set to a low, constant value, so that the cache can shrink if necessary. On Windows platforms, this value is 2 MB.

The *size* is the amount of memory, in bytes. Use `k`, `m`, or `g` to specify units of kilobytes, megabytes, or gigabytes, respectively.

The unit `p` is a percentage either of the physical system memory, or of the maximum non-AWE cache size, whichever is lower. The maximum non-AWE cache size depends on the operating system. For example:

- 2.5 GB for Windows 32-bit Advanced Server, Enterprise Server, and Datacenter Server
- 3.5 GB for the 32-bit database server running on Windows x64 Edition
- 1.5 GB on all other 32-bit systems
- On Windows Mobile, available physical memory limits the cache size is limited by available physical memory
- On 64-bit database servers, the cache size can be considered unlimited

If you use `p`, the argument is a percentage. You can use % as an alternative to `p`, but on Windows, which uses % as an environment variable escape character, you must escape the % character.

For example, to set the minimum cache size to 50 percent of the physical system memory, run the following command:

```
iqsrv15 -cl 50%% ...
```

This option is ignored if you are using an AWE cache.

**Note:** If you attempt to set your initial or minimum cache sizes to a value that is less than one eighth of the maximum cache size, the initial and minimum cache sizes are automatically increased relative to the maximum cache size.

*Example*

The following example starts a database server named silver that has a minimum cache size of 5 MB and loads the database file `example.db`:

```
iqsrv15 -cl 5m -n silver "c:\example.db"
```

## -cm iqsrv15 server option

Specifies the amount of address space allocated for an Address Windowing Extensions (AWE) cache on Windows.

**Note:** The use of AWE is deprecated. It is recommended that you use the 64-bit version of the database server on a 64-bit Windows operating system if you require a large cache.

*Syntax*

```
iqsrv15 -cm { size[ k | m | g | p ] } ...
```

*Applies to*

Windows

*Remarks*

When using an AWE cache on any of the supported platforms, the database server uses its entire address space except for 512 MB to access the cache memory. The 512-MB address space is left available for other purposes, such as DLLs that the server must load and for non-cache memory allocations. On most systems, the default setting is enough. If you need to increase or decrease the amount of reserved address space, you can do so by specifying the -cm option. The database server displays the amount of address space it is using in the database server messages window at startup.

The *size* is the amount of memory, in bytes. Use `k`, `m`, or `g` to specify units of kilobytes, megabytes, or gigabytes, respectively.

The unit `p` is a percentage of the maximum non-AWE cache size. If you use `p`, the argument is a percentage. You can use % as an alternative to `p`, but as most Windows operating systems use

% as an environment variable escape character, you must escape the % character. To set the AWE address space cache size to 50 percent of the address space available to the database server process, you would use the following:

```
iqsrv15 -cm 50%% ...
```

## -cp iqsrv15 server option

Specifies an additional set of directories or JAR files in which to search for classes.

*Syntax*

```
iqsrv15 -cp location[ ;location ... ] ...
```

*Applies to*

All operating systems and database servers.

This option has not been fully tested with servers in the cloud.

*Remarks*

It is recommended that all classes and JAR files that are being used with Java in the database be installed in the database. When you store the classes and JAR files within the database, the database can be easily moved to a different computer or operating system. Another benefit of installing classes and JAR files into the database is that the database server's class loader can fetch the classes and resources from the database, allowing each connection that is using Java in the database to have its own instance of these classes and its own copy of static variables within these classes.

However, when a class or JAR file must be loaded by the system class loader, it can be specified with the java_class_path database option or the -cp iqsrv15 server option. Both options add classes and JAR files to the classpath that the database server builds for launching the Java VM. The java_class_path database option is useful when the server is running multiple databases and each database has a different set of JARs and directories that need to be loaded by the system class loader. The -cp database server option is useful when all databases on the server require the same classes or JAR files.

## -cr iqsrv15 server option

Reloads (warms) the cache with database pages using information collected the last time the database was run.

*Syntax*

```
iqsrv15 -cr{ + | - } ...
```

*Applies to*

All operating systems and database servers.

*Remarks*
You can instruct the database server to warm the cache using pages that were referenced the last time the database was started (page collection is turned on using the -cc option). Cache warming is turned on by default. When a database is started, the server checks the database to see if it contains a collection of pages requested the last time the database was started. If the database contains this information, the previously-referenced pages are then loaded into the cache.

Warming the cache with pages that were referenced the last time the database was started can improve performance when the same query or similar queries are executed against a database each time it is started.

## -cs iqsrv15 server option

Displays statistics related to dynamic cache sizing in the database server messages window.

*Syntax*

```
iqsrv15 -cs ...
```

*Applies to*
Windows, Unix

*Remarks*
For troubleshooting purposes, this option displays statistics in the database server messages window that database server is using to determine how to tune size of the cache.

## -cv iqsrv15 server option

Controls the appearance of messages about cache warming in the database server messages window.

*Syntax*

```
iqsrv15 -cv{ + | - } ...
```

*Default*
Cache warming messages are suppressed.

*Applies to*
All operating systems and database servers.

*Remarks*
When -cv+ is specified, a message appears in the database server messages window when any of the following cache warming activities occur:

- collection of requested pages starts or stops (controlled by the -cc server option)

- page reloading starts or stops (controlled by the -cr server option)

*Example*

The following command starts the database `mydatabase.db` with database page collection and page loading turned on, and logs messages about these activities to the database server messages window:

```
iqsrv15 -cc+ -cr+ -cv+ mydatabase.db
```

## -cw iqsrv15 server option (deprecated)

Enables use of Address Windowing Extensions (AWE) on Windows for setting the size of the database server cache.

**Note:** The use of AWE is deprecated. It is recommended that you use the 64-bit version of the database server on a 64-bit Windows operating system if you require a large cache.

*Syntax*

```
iqsrv15 -cw ...
```

*Applies to*

Windows

*Remarks*

The amount of memory available for use as a database server cache is one of the key factors controlling performance. Because Windows supports Address Windowing Extensions, you can use the -cw option to take advantage of large cache sizes based on the maximum amount of physical memory in the system.

AWE caches are not supported on 64-bit database servers.

| Operating system (32-bit) | Maximum non-AWE cache size | Maximum amount of physical memory supported by Windows |
|---|---|---|
| Windows XP Home Edition | 1.5 GB | 2 GB |
| Windows XP Professional | 1.5 GB | 4 GB |
| Windows Server 2003 Web | 1.5 GB | 2 GB |
| Windows Server 2003 Standard | 1.5 GB | 4 GB |
| Windows Server 2003 Enterprise | 2.5 GB[1] | 32 GB |
| Windows Server 2003 Datacenter | 2.5 GB[1] | 64 GB |

| **Operating system (32-bit)** | **Maximum non-AWE cache size** | **Maximum amount of physical memory supported by Windows** |
|---|---|---|
| Windows Vista Ultimate | 2.5 GB[1] | 4 GB |
| Windows Vista Enterprise | 2.5 GB[1] | 4 GB |
| Windows Vista Business | 2.5 GB[1] | 4 GB |
| Windows Vista Home Premium | 2.5 GB[1] | 4 GB |
| Windows Vista Home Basic | 2.5 GB[1] | 4 GB |
| Windows Vista Starter | 2.5 GB[1] | 1 GB |
| Windows 7 Ultimate | 2.5 GB[1] | 4 GB |
| Windows 7 Enterprise | 2.5 GB[1] | 4 GB |
| Windows 7 Professional | 2.5 GB[1] | 4 GB |
| Windows 7 Home Premium | 2.5 GB[1] | 4 GB |
| Windows 7 Home Basic | 2.5 GB[1] | 4 GB |
| Windows 7 Starter | 2.5 GB[1] | 1 GB |
| Windows Server 2008 Datacenter | 2.5 GB[1] | 64 GB |
| Windows Server 2008 Enterprise | 2.5 GB[1] | 64 GB |
| Windows Server 2008 Standard | 2.5 GB[1] | 4 GB |
| Windows Server 2008 Small Business Server | 2.5 GB[1] | 4 GB |
| Windows Web Server 2008 | 2.5 GB[1] | 4 GB |

[1] If you have 2 GB-16 GB of memory, enable 4 gigabyte tuning (4GT):

- **Windows XP/2003 –** Add the `/3GB` option to the Windows boot line in the "[operating systems]" section of the `boot.ini` file.
- **Windows Vista and later –** Run the following command as an administrator:

```
bcdedit /set increaseuserva 3072
```

When using an AWE cache, most of the available physical memory in the system can be allocated for the cache.

If you can set a cache of the required size using a non-AWE cache, this is recommended because AWE caches allocate memory that can only be used by the database server. This means that while the database server is running, the operating system and other applications cannot use the memory allocated for the database server cache. AWE caches do not support dynamic cache sizing. Therefore, if an AWE cache is used and you specify the -ch or -cl options to set the upper and lower cache size, they are ignored.

By default, 512 MB of address space is reserved for purposes other than the database server's AWE cache (address space is the amount of memory that can be accessed by a program at any given time). Although this amount is usually enough, you can use the -cm option to change the amount of reserved address space.

On Windows Vista and Windows 7, only elevated database servers can use AWE memory. If you are starting a database server automatically on Windows Vista or Windows 7, you must specify ELEVATE=YES in your connection string so that the database server executables are elevated.

To start a database server with an AWE cache, you must do the following:

- On Windows Vista and Windows 7, you must run the database server as an administrator.
- Have at least 130 MB of memory available on your system.
- If you have 2 GB-16 GB of memory, enable 4 gigabyte tuning (4GT):
  - **Windows XP/2003 –** Add the `/3GB` option to the Windows boot line in the "[operating systems]" section of the `boot.ini` file.
  - **Windows Vista and later –** Run the following command as an administrator:

    ```
    bcdedit /set increaseuserva 3072
    ```
- If you have more than 16 GB of memory, do not enable 4GT.
- If you have at least 4 GB of memory, enable PAE:
  - **Windows XP/2003 –** Add the `/PAE` option to the Windows boot line in the [operating systems] section of the `boot.ini` file.
  - **Windows Vista and later –** Run the following command as an administrator:

    ```
    bcdedit /set pae ForceEnable
    ```
- Grant the Lock Pages in Memory privilege to the user ID under which the server is run. The following steps explain how to do this on Windows XP.
  1. Log on to Windows as an administrator.
  2. Open the **Control Panel**.
  3. Double-click **Administrative Tools**.
  4. Double-click **Local Security Policy**.
  5. Click **Local Policies** in the left pane.
  6. Double-click **User Rights Assignment**.
  7. Double-click the **Lock Pages In Memory** policy in the right pane.
  8. Click **Add User Or Group**.
  9. Type the name of the user, and then click **OK**.
  10. Click **OK** on the **Lock Pages In Memory** window.
  11. Close all open windows and restart the computer for the setting to take effect.

If you specify the -cw option and the -c option, the database server attempts the initial cache allocation as follows:

1. The AWE cache is no larger than the cache size specified by the -c option. If the value specified by the -c option is less than 2 MB, AWE isn't used.
2. The AWE cache is no larger than all available physical memory less 128 MB.
3. The AWE cache is no smaller than 2 MB. If this minimum amount of physical memory isn't available, an AWE cache isn't used.

When you specify the -cw option and do not specify the -c option, the database server attempts the initial cache allocation as follows:

1. The AWE cache uses 100% of all available memory except for 128 MB that is left free for the operating system.
2. The AWE cache is no larger than the sum of the sizes of the main database files specified on the command line. Additional dbspaces apart from the main database files aren't included in the calculation. If no files are specified, this value is zero.
3. The AWE cache is no smaller than 2 MB. If this minimum amount of physical memory isn't available, an AWE cache isn't used.

When the server uses an AWE cache, the cache page size will be no smaller than 4 KB and dynamic cache sizing is disabled.

*Example*

The following example starts a database server named `myserver` that starts with a cache size of 12 GB and loads the database `c:\test\mydemo.db`:

```
iqsrv15 -n myserver -c 12G -cw c:\test\mydemo.db
```

## AWE Cache Sizes per Operating System

Use this table as a reference if using the **start_iq** server option **-cw** to use Address Windowing Extensions (AWE) when setting the size of the catalog store cache.

| Operating system | Maximum non-AWE cache size | Maximum amount of physical memory supported by Windows |
|---|---|---|
| Windows 2000 Professional | 1.8GB | 4GB |
| Windows 2000 Server | 1.8GB* | 4GB |
| Windows 2000 Advanced Server | 2.7GB* | 8GB |
| Windows 2000 Datacenter Server | 2.7GB* | 64GB |
| Windows XP Home Edition | 1.8GB | 2GB |
| Windows XP Professional | 1.8GB | 4GB |

| Operating system | Maximum non-AWE cache size | Maximum amount of physical memory supported by Windows |
|---|---|---|
| Windows Server 2003, Web Edition | 1.8GB | 2GB |
| Windows Server 2003, Standard Edition | 1.8GB | 4GB |
| Windows Server 2003, Enterprise Edition | 2.7GB* | 32GB |
| Windows Server 2003, Datacenter Edition | 2.7GB* | 64GB |

*You must restart the operating system using the `/3GB` option to use a cache of this size.

### See also

- *Initial Catalog Store Cache Size when start_iq Server Option -c is not Specified* on page 117
- *Database Server Naming Restrictions* on page 157
- *AWE Cache Allocation* on page 128
- *Starting a Database Server with an AWE Cache* on page 129

## AWE Cache Allocation

The **-cw** and **-c** server options affect the initial cache allocation.

If you specify **start_iq** server options **-cw** and **-c** on the command line, the database server attempts the initial cache allocation as follows:

- The AWE cache is no larger than the cache size specified by the **-c** option. If the value specified by the **-c** option is less than 2MB, AWE is not used.
- The AWE cache is no larger than all available physical memory less 128MB.
- The AWE cache is no smaller than 2MB. If this minimum amount of physical memory is not available, an AWE cache is not used.

When you specify the **-cw** option and do not specify the **-c** option, the database server attempts the initial cache allocation as follows:

- The AWE cache uses 100% of all available memory except for 128MB that is left free for the operating system.
- The AWE cache is no larger than the sum of the sizes of the main database files specified on the command line. Additional dbspaces apart from the main database files are not included in the calculation. If no files are specified, this value is zero.

- The AWE cache is no smaller than 2MB. If this minimum amount of physical memory is not available, an AWE cache is not used.

When the server uses an AWE cache, the catalog cache page size is at least 4KB, and dynamic cache sizing is disabled. On 64-bit Windows platforms, the cache page size is at least 8KB.

For more information about dynamic cache sizing, see the **-ch** and **-cl** server options.

### See also

- *start_iq Server Options* on page 115
- *AWE Cache Sizes per Operating System* on page 127
- *Starting a Database Server with an AWE Cache* on page 129

## Starting a Database Server with an AWE Cache

Start a database with an AWE cache to take advantage of the large cache sizes available with Windows 2000, Windows XP, and Windows Server 2003. You can instruct Sybase IQ to use AWE using the **start_iq** server option **-cw**.

### Prerequisites

Prerequisites:

- You must have at least 130MB of memory available on your system.
- If your system has between 2GB and 16GB of memory, add the `/3GB` option to the Windows boot line in the `boot.ini` file.

  If your system has more than 16GB of memory, do not add the `/3GB` option to the Windows boot line in the `boot.ini` file because Windows cannot address memory beyond 16GB.
- If your system has more than 4GB of memory, add the `/PAE` option to the Windows boot line in the `boot.ini` file.

### Task

1. Log in to Windows as Administrator
2. Select **Start | Settings | Control Panel.**
3. Open the Administrative Tools folder.
4. Double-click **Local Security Policy.**
5. In the left pane, open **Local Policies.**
6. In the left pane, double-click **User Rights Assignment.**
7. In the right pane, double-click the **Lock Pages In Memory** policy.
8. In the Local Security Policy Setting dialog, click **Add**.
9. Select the user ID and click **Add**.

10. In the Local Security Policy Setting dialog, click **OK**.
11. Restart the computer.

**See also**

- *Initial Catalog Store Cache Size when start_iq Server Option -c is not Specified* on page 117
- *Database Server Naming Restrictions* on page 157
- *AWE Cache Allocation* on page 128
- *start_iq Server Options* on page 115
- *AWE Cache Sizes per Operating System* on page 127

## -dt iqsrv15 server option

Specifies the directory where temporary files are stored.

*Syntax*

```
iqsrv15 -dt temp-file-dir ...
```

*Applies to*

All servers and operating systems, except shared memory connections on Unix.

*Remarks*

The database server creates two types of temporary files:

- **Database server-related temporary files (created on all platforms) –** You can use the -dt option to specify a directory for database server-related temporary files. If you do not specify this option when starting the database server, the database server examines the following environment variables, in the order shown, to determine the directory in which to place the temporary file.

  | Windows/Unix |
  |---|
  | 1. IQTMP15<br>2. TMP<br>3. TMPDIR<br>4. TEMP |

  If none of the environment variables are defined, the database server places its temporary file in the current directory on Windows, and in the `/tmp` directory on Unix.

  On Windows Mobile, the `TempFolder` registry key entry is used to determine the directory for temporary files. If it is not defined, `\Temp` is used. If the directory does not exist and cannot be created, the current directory is used.
- **Communications-related temporary files (created only on Unix for both the client and the database server) –** Temporary files for communications on Unix are not placed in

the directory specified by -dt. Instead, the database server examines the following environment variables, in the order shown, to determine the directory in which to place the temporary file:

| Unix |
|---|
| **1.** IQTMP15 |
| **2.** TMP |
| **3.** TMPDIR |
| **4.** TEMP |

If none of the environment variables are defined, the database server places its temporary file in the `/tmp` directory on Unix.

On Unix, both the client and the database server must set IQTMP15 to the same value when connecting via shared memory.

*Example*

To locate the database server-related temporary files, use the DB_PROPERTY system function with the TempFileName property:

```
SELECT DB_PROPERTY ( 'TempFileName' );
```

## -ec iqsrv15 server option

Uses transport-layer security or simple encryption to encrypt all command sequence communication protocol packets (such as DBLib and ODBC) transmitted to and from all clients. TDS packets aren't encrypted.

*Syntax*

```
iqsrv15 -ec encryption-options ...

encryption-options :

{ NONE |
   SIMPLE |
   TLS ( TLS_TYPE=algorithm;
   [ FIPS={ Y | N }; ]
   IDENTITY=server-identity-filename;
   IDENTITY_PASSWORD=password ) }, ...
```

*Allowed values*

- **NONE –** accepts connections that aren't encrypted.
- **SIMPLE –** accepts connections that are encrypted with simple encryption. This type of encryption is supported on all platforms, and on previous versions of the database serverand clients. Simple encryption doesn't provide server authentication, RSA encryption, or other features of transport-layer security.

- **TLS –** accepts connections that are encrypted. The TLS parameter accepts the following required arguments:
  - **algorithm –** can be RSA for RSA encryption. For FIPS-certified RSA encryption, specify **TLS_TYPE=RSA;FIPS=Y**. RSA FIPS-certified encryption uses a separate certified library, but is compatible with version 9.0.2 or later clients specifying RSA.

    The algorithm must match the RSA encryption used to create your certificates.
  - **server-identity-filename –** is the path and file name of the server identity certificate. If you are using FIPS-certified RSA encryption, you must generate your certificates using the RSA algorithm.
  - **password –** is the password for the server private key. You specify this password when you create the server certificate.

*Applies to*

NONE and SIMPLE apply to all servers and operating systems.

TLS applies to all servers and operating systems.

*Remarks*

You can use this option to secure communication packets between client applications and the database server using transport-layer security.

**Note:** Separately licensed component required.

FIPS-certified encryption requires a separate license. All strong encryption technologies are subject to export regulations.

The -ec option instructs the database server to accept only connections that are encrypted using one of the specified types. You must specify at least one of the supported parameters in a comma-separated list. Connections over the TDS protocol, which include Java applications using jConnect, are always accepted and are never encrypted, regardless of the usage of the -ec option. Setting the TDS protocol option to NO disallows these unencrypted TDS connections.

By default, communication packets aren't encrypted, which poses a potential security risk. If you are concerned about the security of network packets, use the -ec option. Encryption affects performance only marginally.

If the database server accepts simple encryption, but does not accept unencrypted connections, then any non-TDS connection attempts using no encryption automatically use simple encryption.

Starting the database server with **-ec SIMPLE** tells the database server to only accept connections using simple encryption. TLS connections (RSA and RSA FIPS-certified encryption) fail, and connections requesting no encryption use simple encryption.

The `dbrsa12.dll` file contains the RSA code used for encryption and decryption. The file `dbfips12.dll` contains the code for the FIPS-certified RSA algorithm. When you connect to the database server, if the appropriate file cannot befound, or if an error occurs, a message appears in the database server messages window. The server doesn't start if the specified types of encryption cannot be initiated.

The client's and the server's encryption settings must match or the connection fails except in the following cases:

- If -ec SIMPLE is specified on the database server, but -ec NONE is not, then connections that do not request encryption can connect and automatically use simple encryption.
- If the database server specifies RSA and the client specifies FIPS-certified encryption, or vice versa, the connection succeeds. In these cases, the Encryption connection property returns the value specified by the database server.

*Example*

The following example specifies that connections with no encryption and simple encryption are allowed.

```
iqsrv15 -ec NONE,SIMPLE -x tcpip c:\mydemo.db
```

The following example starts a database server that uses the RSA server certificate `rsaserver.id`.

```
iqsrv15 -ec
TLS(TLS_TYPE=RSA;IDENTITY=rsaserver.id;IDENTITY_PASSWORD=test) -x
tcpip c:\mydemo.db
```

The following example starts a database server that uses the FIPS-approved RSA server certificate `rsaserver.id`.

```
iqsrv15 -ec
TLS(TLS_TYPE=RSA;FIPS=Y;IDENTITY=rsaserver.id;IDENTITY_PASSWORD=tes
t) -x tcpip c:\mydemo.db
```

## -ep iqsrv15 server option

Prompts the user for the encryption key upon starting a strongly encrypted database.

*Syntax*

```
iqsrv15 -ep ...
```

*Applies to*

All operating systems and database servers.

This option does not apply to servers in the cloud.

*Remarks*

The -ep option instructs the database server to display a window where the user enters the encryption key for databases started on the command line that require an encryption key. This server option provides an extra measure of security by never allowing the encryption key to be seen in clear text.

When used with the server, the user is prompted for the encryption key when the following are all true:

- the -ep option is specified
- the server is a Windows personal server, or the server is just starting up
- a key is required to start a database
- the database server is either not a Windows service, or it is a Windows service with the interact with desktop option turned ON
- the server isn't a daemon (Unix)

If you want to secure communication packets between client applications and the database server use the -ec server option and transport-layer security.

*Example*

The user is prompted for the encryption key when the `myencrypted.db` database is started:

```
iqsrv15 -ep -x tcpip myencrypted.db
```

## -es iqsrv15 server option

Allows unencrypted connections over shared memory.

*Syntax*

```
iqsrv15 -ec encryption-options -es ...
```

*Applies to*

All servers and operating systems, except Windows Mobile.

This option does not apply to servers in the cloud.

*Remarks*

This option is only effective when specified with the -ec option. The -es option instructs the database server to allow unencrypted connections over shared memory. Connections over TCP/IP must use an encryption type specified by the -ec option. This option is useful in situations where you want remote clients to use encrypted connections, but for performance reasons you also want to access the database from the local computer with an unencrypted connection.

*Example*

The following example specifies that connections with simple encryption and unencrypted connections over shared memory are allowed.

```
iqsrv15 -ec SIMPLE -es -x tcpip c:\mydemo.db
```

## -fips iqsrv15 server option

Requires that only FIPS-certified algorithms can be used for strong database and communication encryption.

*Syntax*

```
iqsrv15 -fips ...
```

*Applies to*

Windows, Unix, and Linux.

*Remarks*

Specifying this option forces all database server encryption to use FIPS-certified algorithms. This option applies to strong database encryption, client/server transport-layer security, and web services transport-layer security. You can still use unencrypted connections and databases when the -fips option is specified, but you cannot use simple encryption.

**Note:** Separately licensed component required.

FIPS-certified encryption requires a separate license. All strong encryption technologies are subject to export regulations.

For strong database encryption, the -fips option causes new databases to use the FIPS-certified equivalent of AES and AES256 if they are specified in the **ALGORITHM** clause of the **CREATE DATABASE** statement.

When the database server is started with -fips, you can run databases encrypted with AES, AES256, AES_FIPS, or AES256_FIPS encryption, but not databases encrypted with simple encryption. Unencrypted databases can also be started on the server when -fips is specified.

The security option must be installed on any computer used to run a database encrypted with AES_FIPS or AES256_FIPS.

For transport-layer security, the -fips option causes the server to use the FIPS-certified RSA encryption algorithm, even if RSA is specified.

For transport-layer security for web services, the -fips option causes the server to use FIPS-certified HTTPS, even if HTTPS is specified.

When you specify -fips, the **ENCRYPT** and **HASH** functions use the FIPS-certified RSA encryption algorithm, and password hashing uses the SHA-256 FIPS algorithm rather than the SHA-256 algorithm.

## -ga iqsrv15 server option

Unloads the database after the last non-HTTP client connection disconnects.

*Syntax*

```
iqsrv15 -ga ...
```

*Applies to*

All operating systems.

This option does not apply to servers in the cloud.

*Remarks*

Specifying this option on the network server causes each database to be unloaded after the last non-HTTP client connection disconnects. In addition to unloading each database after the last non-HTTP connection disconnects, the database server shuts down when the last database is stopped.

If the only connection to a database is an HTTP connection, and the database is configured to stop automatically, when the HTTP connection disconnects, the database is not unloaded. As well, if you specify the -ga option, and the database has an HTTP connection and a command sequence or TDS connection, when the last command sequence or TDS connection disconnects, the database stops automatically, and any HTTP connections are dropped.

## -gb iqsrv15 server option

Sets the server process priority class.

*Windows syntax*

```
iqsrv15 -gb { idle | normal | high | maximum } ...
```

*Unix syntax*

```
iqsrv15 -gb level ...
```

*Allowed values*

- **Unix –** On Unix, the *level* is an integer from -20 to 19. The default value on Unix is the same as the nice value of the parent process. Lower *level* values represent a more favorable scheduling priority. All restrictions placed on setting a nice value apply to the -gb option. For example, on most Unix platforms, only the root user can lower the priority level of a process (for example, changing it from 0 to -1).
- **Windows –** On Windows, normal and high are the commonly-used settings. The value idle is provided for completeness, and maximum may interfere with the running of your computer.

*Applies to*
Windows, Unix.

This option has not been fully tested with servers in the cloud.

## -gc iqsrv15 server option

Sets the maximum interval between checkpoints.

*Syntax*

```
iqsrv15 -gc minutes ...
```

*Default*
60 minutes

*Allowed values*

- **minutes –** The default value is the setting of the checkpoint_time database option, which defaults to 60 minutes. If a value of 0 is entered, the default value of 60 minutes is used.

*Applies to*
All operating systems and database servers.

*Remarks*
Use this option to set the maximum length of time, in minutes, that the database server runs without doing a checkpoint on each database.

Checkpoints generally occur more frequently than the specified time.

## -gd iqsrv15 server option

Sets the permissions required to start or stop a database on a running database server.

*Syntax*

```
iqsrv15 -gd { DBA | all | none } ...
```

*Allowed values*

- **DBA –** Only users with DBA authority can start or stop databases.
- **all –** All users can start or stop databases. Not recommended for network servers that can be accessed by remote clients.
- **none –** Starting and stopping databases isn't allowed except when the database server itself is started and stopped.

*Default*

The default setting is `all` for the personal database server, and `DBA` for the network database server. Both uppercase and lowercase syntax are allowed.

*Applies to*

All operating systems and database servers.

This option does not apply to servers in the cloud.

*Remarks*

This option specifies the level of permission required for a user to cause a new database file to be loaded by the database server, or to stop a database on a running database server.

When the option is set to DBA, the client application must use an existing DBA connection to another database running on the same server to start and stop databases. Simply having a connection to the server is not enough.

You can obtain the setting of the -gd option using the StartDBPermission server property:

```
SELECT PROPERTY ( 'StartDBPermission' );
```

The permissions for stopping a database server are specified by the -gk option.

*Example*

The following steps illustrate how to use the -gd option for the network database server.

1. Start the network database server:
   ```
   iqsrv15 -su mypwd -gd DBA -n my_server
   ```
2. Connect to the utility database from Interactive SQL:
   ```
   dbisql -c "UID=DBA;PWD=mypwd;DBN=utility_db"
   ```
3. Start a database:
   ```
   START DATABASE 'iqdemo.db';
   ```
4. Connect to the database:
   ```
   CONNECT USING 'DBN=demo;UID=DBA;PWD=sql';
   ```

## -ge iqsrv15 server option

Sets the stack size for external functions.

*Syntax*

```
iqsrv15 -ge integer ...
```

*Default*

32 KB

*Applies to*

Windows.

This option has not been fully tested with servers in the cloud.

*Remarks*

Sets the stack size for threads running external functions, in bytes.

## -gf iqsrv15 server option

Disables firing of triggers by the server.

*Syntax*

```
iqsrv15 -gf ...
```

*Applies to*

All operating systems and database servers.

This option does not apply to servers in the cloud.

*Remarks*

The -gf server option instructs the server to disable the firing of triggers.

## -gk iqsrv15 server option

Sets the permissions required to stop the network server and personal server using dbstop.

*Syntax*

```
iqsrv15 -gk { DBA | all | none } ...
```

*Allowed values*

- **DBA –** Only users with DBA authority can use dbstop to stop the database server. This is the default for the network server.
- **all –** No permissions are required to shut down the database server. This is the default for the personal server.
- **none –** The database server cannot be stopped using dbstop.

*Applies to*

All operating systems and database servers.

This option does not apply to servers in the cloud.

## -gl iqsrv15 Server Option

Set the permission required to load data using **LOAD TABLE**.

*Syntax*

**-gl level**

*Remarks*

The **LOAD TABLE** statement reads files from the database server machine. To control access to the file system using these statements, the **-gl** command-line switch allows you to control the level of database permission that is required to use these statements. *level* is either:

- DBA—only users with DBA authority can load data.
- ALL—all users can load data.
- NONE—data cannot be loaded.

You can use either uppercase and lowercase syntax for the options.

The default settings are **all** for servers started with **start_iq** and **dba** for other servers. Sybase recommends that, for consistency with earlier versions, you use the **all** value on all systems. The **all** setting is used in the `iqdemo.cfg` and `default.cfg` configuration files.

## -gm iqsrv15 server option

Limits the number of concurrent connections to the database server.

*Syntax*

```
iqsrv15 -gm integer ...
```

*Default*

The default value for the personal server is 10. The default value for the network database server is 32766, though this number will be reduced by internal temporary connections utilized by the server during operation.

*Applies to*

All operating systems and database servers.

This option has not been fully tested with servers in the cloud.

*Remarks*

Defines the connection limit for the server. If this number is greater than the number that is allowed under licensing and memory constraints, it has no effect. Computer resources typically limit the number of connections to a network server to a lower value than the default.

The database server allows one extra DBA connection above the connection limit to allow a user with DBA authority to connect to the database server and drop other connections.

## -gn iqsrv15 Server Option

Sets the number of execution threads that will be used for the catalog store and connectivity while running with multiple users.

*Syntax*

**-gn integer**

*Remarks*
This parameter applies to all operating systems and servers. Each connection uses a thread for each request, and when the request is completed, the thread is returned to the pool for use by other connections. As no connection can have more than one request in progress at one time, no connection uses more than one thread at a time.

An exception to this rule is if a Java application uses threads. Each thread in the Java application is a database server execution thread.

On Windows, specify this parameter in **start_iq**. To calculate its value, use:

```
gn_value >= gm_value * 1.5
```

Sybase recommends that you set the **-gn** value to at least 1.5 times the value of **-gm**. Specify a minimum of 25. The total number of threads must not exceed a platform-specific maximum; see **-iqmt num** for details.

The **start_iq** utility sets this parameter. See the *Installation and Configuration Guide* for your platform for more information.

## -gp iqsrv15 server option

Sets the maximum allowed database page size.

*Syntax*

```
iqsrv15 -gp { 2048 | 4096 | 8192 | 16384 | 32768 } ...
```

*Default*
4096 (if a database server is started with no databases loaded)

*Applies to*
All operating systems and database servers.

This option does not apply to servers in the cloud.

*Remarks*
Database files with a page size larger than the page size of the server cannot be loaded. This option explicitly sets the page size of the server, in bytes.

By default, the server page size is the same as the largest page size of the databases on the command line.

On all platforms, if you do not use this option and start a server with no databases loaded, the default value is 4096.

## -gr iqsrv15 Server Option

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

*Syntax*

**-gr** *num*

*Remarks*

See *Reference: Statements and Options > Database Options > Alphabetical List of Options > RECOVERY_TIME option*.

## -gss iqsrv15 Server Option

Sets, in part, the stack size for server execution threads that execute requests for server connections.

*Syntax*

**-gss** {*integer* | *integerK* | *integerM*}

*Remarks*

IQ calculates the stack size of these server threads using the formula: (**-gss** + **-iqtss).** See **-iqtss**.

On Windows, the default value for **-gss** is 4MB. The maximum stack size is 256MB.

## -gt iqsrv15 server option

Sets the maximum number of physical processors that can be used (up to the licensed maximum). This option is only useful on multiprocessor systems.

*Syntax*

```
iqsrv15 -gt num-processors ...
```

*Allowed values*

- **num-processors –** This integer can be a value between 1 and the minimum of:

  the number of physical processors on the computer
  the maximum number of CPUs that the server is licensed for if CPU-licensing is in effect

If the -gt value specified lies outside this range, the lower or upper limit is imposed. For the personal database server, the server uses a -gt value of 1.

*Applies to*

Windows (except Windows Mobile), Linux, and Solaris.

*Remarks*

The personal database server is always limited to a single processor. With per-seat licensing, the network database server uses all CPUs available on the computer (the default). With CPU-based licensing, the network database server uses only the number of processors you are licensed for. The number of CPUs that the network database server can use may also be affected by your SQL Anywhere edition.

When you specify a value for the -gt option, the database server adjusts its affinity mask (if supported on that hardware platform) to restrict the database server to run on only that number of physical processors. If the database server is licensed for *n* processors, the server, by default, runs on all logical processors (hyperthreads and cores) of *n* physical processors. This behavior can be further restricted with the -gtc option.

## -gtc iqsrv15 server option

Controls the maximum processor concurrency that the database server allows.

*Syntax*

```
iqsrv15 -gtc logical-processors-to-use ...
```

*Applies to*

Linux, Solaris, and Windows operating systems executing on Intel-compatible x86 and x64 platforms, excluding Windows Mobile.

*Remarks*

When you start the database server, the number of physical and logical processors detected by the database server appears in the database server messages window.

Physical processors are sometimes referred to as **packages** or **dies**, and are the CPUs of the computer. Additional logical processors exist when the physical processors support hyperthreading or are themselves configured as **multiprocessors** (usually referred to as multi-core processors). The operating system schedules threads on logical processors.

The -gtc option allows you to specify the number of logical processors that can be used by the database server. Its effect is to limit the number of database server threads that are created at server startup. This limits the number of active database server tasks that can execute concurrently at any one time. By default, the number of threads created is 1 + the number of logical processors on all licensed physical processors.

By default, the database server allows concurrent use of all logical processors (cores or hyperthreads) on each licensed physical processor. For example, on a single-CPU system that supports hyperthreading, by default the database server permits two threads to run concurrently on one physical processor. If the -gtc option is specified, and the number of logical processors to be used is less than the total available for the number of physical processors that are licensed, then the database server allocates logical processors based on round-robin assignment. Specifying 1 for the -gtc option implicitly disables intra-query parallelism (parallel processing of individual queries). Intra-query parallelism can also be explicitly limited or disabled outright using the max_query_tasks option.

*Example*

Consider the following examples for a Windows-based SMP computer. In each case, assume a 4-processor system with two cores on each physical processor for a total of eight logical processors. The physical processors are identified with letters and the logical processors (cores in this case) are identified with numbers. This 4-processor system therefore has processing units A0, A1, B0, B1, C0, C1, D0, and D1.

| Scenario | Network database server settings |
|---|---|
| A single CPU license or -gt 1 specified | -gt 1<br>-gtc 2<br>-gnh 20<br><br>Threads can execute on A0 and A1. |
| No licensing restrictions on the CPU with -gtc 5 specified | -gt 4<br>-gtc 5<br>-gnh 20<br><br>Threads can execute on A0, A1, B0, C0, and D0. |
| A database server with a three CPU license and -gtc 5 specified | -gt 3<br>-gtc 5<br>-gnh 20<br><br>Threads can execute on A0, A1, B0, B1, and C0. |
| No licensing restrictions on the CPU with -gtc 1 specified | -gt 4<br>-gtc 1<br>-gnh 20<br><br>Threads can execute only on A0. |

## -gu iqsrv15 server option

Sets the permission levels for utility commands.

*Syntax*

```
iqsrv15 -gu { all | none | DBA | utility_db } ...
```

*Default*
DBA

*Applies to*
All operating systems and database servers.

This option does not apply to servers in the cloud.

*Remarks*
Sets permission levels for utility commands such as CREATE DATABASE and DROP DATABASE.

The utility_db level restricts the use of these commands to only those users who can connect to the utility database.

The value `all` is deprecated.

## -iqfrec iqsrv15 Server Option

Open database in forced recovery mode.

*Syntax*
**-iqfrec** *dbname*

## -iqgovern iqsrv15 Server Option

Sets the number of concurrent queries allowed by the server.

*Syntax*
**-iqgovern** *num*

*Remarks*
The number of concurrent queries is not the same as the number of connections. A single connection can have multiple open cursors. **-iqgovern** can help Sybase IQ optimize paging of buffer data out to disk and avoid overcommitting memory. The default value of this switch is equal to 2 times the number of CPUs on your machine, plus 10. You may find that another value, such as 2 times the number of CPUs plus 4, provides better throughput, especially when large numbers of users are connected.

## -iqmc iqsrv15 Server Option

Specifies the main IQ store cache size, in MB.

*Syntax*
**-iqmc** *size*

*Remarks*

Always specify the value for the size, but no units of measurement; for example specify `32` instead of `32MB`. If you specify the unit of measurement, **start_iq** ignores this switch, unlike SQL Anywhere, which requires a unit of measurement.

The switch overrides default of 64MB. Applies to all databases started from the time the IQ server is started until the IQ server is shut down. In other words, if you start one database at server startup and another later, you need 2 * **-iqmc** available for the main cache. In general, Sybase recommends that you do not run multiple databases with a Sybase IQ server.

## -iqmpx_failover iqsrv15 Server Option

Initiates multiplex coordinator failover to establish the designated failover Secondary node as the new coordinator.

*Remarks*

Starting the coordinator with this option has no effect. Users must be licensed for the Multiplex Grid Option to run secondary nodes. For `iqmpx_failover` values, see *Using Sybase IQ Multiplex*.

## -iqmpx_ov iqsrv15 Server Option

Performs multiplex configuration override for the current node.

*Remarks*

Used to change node properties during startup in the event that a node's location or other property has changed. Users must be licensed for the Multiplex Grid Option to run secondary nodes. For `iqmpx_ov` values, see *Using Sybase IQ Multiplex*.

## -iqmpx_reclaimwriterfreelist iqsrv15 Server Option

This option applies only while restarting a coordinator node.

*Remarks*

The coordinator will forcefully reclaim the free-list of the writer node identified by server-name. This switch is only needed in the event that a writer fails and cannot be restarted. Users must be licensed for the Multiplex Grid Option to run secondary nodes. For `iqmpx_reclaimwriterfreelist` values, see *Using Sybase IQ Multiplex*.

## -iqmpx_sn iqsrv15 Server Option

Runs the current node in multiplex in single node mode.

*Remarks*

This mode is used exclusively for fixing problems with the multiplex configuration and should be used with extreme caution. Requires all other nodes in the multiplex to be shut down.

Recommended only for use on the coordinator node. Users must be licensed for the Multiplex Grid Option to run secondary nodes. For `iqmpx_sn` values, see *Using Sybase IQ Multiplex*.

## -iqmsgnum iqsrv15 Server Option

Specifies the number of archives of the old message log maintained by the server.

### *Syntax*

**-iqmsgnum** *num*

### *Remarks*

Allowed values are integers 0 – 64 (inclusive). Default value is 0, which means that messages are wrapped in the main message log file. Takes effect only if **-iqmsgsz** or the **IQMsgMaxSize** server property is a value other than zero. The **IQMsgNumFiles** server property corresponds to **-iqmsgnum** and takes precedence over the value of **-iqmsgnum**.

A **-iqmsgnum** value *n* greater than 0 means that the server maintains *n* message log archives. For example, this command specifies that the server maintain 3 archives of the message log:

```
start_iq -n iqdemo iqdemo.db ... <other options> ... -iqmsgsz 100 -
iqmsgnum 3
```

See *Reference: Building Blocks, Tables, and Procedures > System Procedures > Catalog Stored Procedures > sa_server_option System Procedure*.

See *System Administration Guide: Volume 1 > Overview of Sybase IQ System Administration > Message Log Contents*.

## -iqmsgsz iqsrv15 Server Option

Limits the maximum size of the message log.

### *Syntax*

**-iqmsgsz** *size*

### *Remarks*

**-iqmsgsz** is an integer 0-2047 (inclusive), in MB. The default value is 0, which specifies that there is no limit on message log size. The **IQMsgMaxSize** server property corresponds to the **-iqmsgsz** server switch and takes precedence over the value of **-iqmsgsz**.

A **-iqmsgsz** value *n* greater than 0 means that the message log can grow up to *n* megabytes in size. For example, the following command limits the size of the message log to 100MB:

```
start_iq -n iqdemo iqdemo.db ... <other options> ... -iqmsgsz 100
```

See *Reference: Building Blocks, Tables, and Procedures > System Procedures > Catalog Stored Procedures > sa_server_option System ProcedureReference: Building Blocks, Tables, and Procedures > System Procedures > sa_server_option system procedure*.

See *System Administration Guide: Volume 1 > Overview of Sybase IQ System Administration > Message Log Contents*.

## -iqmt iqsrv15 Server Option

Specifies the number of Sybase IQ threads to create.

### *Syntax*

**-iqmt** *num*

### *Remarks*

The default is 60 threads per CPU for the first 4 CPUs and 50 threads per CPU for the remainder, with 3 more for system use, plus threads needed for database connections and background tasks. For example, on a system with 12 CPUs and 10 connections: `60*4 + 50*(numCPUs - 4) + numConnections + 6 = 656`.

The minimum value of num is `num_conn + 3`.

The total number of server threads cannot exceed 4096 on 64-bit platforms, or 2048 on 32-bit platforms.

## -iqnotemp iqsrv15 Server Option

Creates a temporary file in place of the defined temporary dbspace.

### *Syntax*

**-iqnotemp size**

### *Remarks*

*size* is file size, in MB. This parameter causes the server to ignore all temporary IQ dbfile definitions when starting a database. You can use **-iqnotemp** to solve temporary dbfile problems by dropping damaged files and replacing them later.

If you start the utility database server with **-iqnotemp** when restoring, Sybase IQ ignores all temporary IQ file definitions in the backed up database during the restore operation, including intermediate starts. You can thus restore a database to a different temporary file topology without recreating and using old temporary file definitions.

The only temporary file operation allowed on a database started with **-iqnotemp** is **ALTER DBSPACE IQ_SYSTEM_TEMP DROP FILE**.

## -iqnumbercpus iqsrv15 Server Option

Specifies the number of CPUs available to IQ, overriding the physical number of CPUs, for resource planning purposes.

*Syntax*

**-iqnumbercpus** *num*

*Remarks*

The value of **-iqnumbercpus** defaults to the total number of CPUs, but the range of available values is 1 – 128.

Sybase recommends that you use **-iqnumbercpus** only on:

- Machines with Intel CPUs and hyperthreading enabled, setting **-iqnumbercpus** to the number of CPU cores available
- Machines where an operating system utility has been used to restrict Sybase IQ to a subset of the CPUs within the machine

Setting **-iqnumbercpus** higher than the number of available CPUs may affect performance.

## -iqpartition iqsrv15 Server Option

Sets the number of IQ main and temp buffer cache partitions.

*Syntax*

**-iqpartition** *num*

*Remarks*

Specifies the number of partitions in the IQ main and temp buffer caches. Must be a power of 2. Allowed values are: 0 (default), 1, 2, 4, 8, 16, 32, 64, 128, 256. By default, IQ computes the number of partitions automatically as *number_of_cpus/8*, rounded to the nearest power of 2, up to a maximum of 64. You may be able to improve performance by adjusting the number of cache partitions. The **-iqpartition** switch sets this value for an IQ server, and overrides the value set by the `Cache_Partitions` database option.

See *Reference: Statements and Options> Database Options > Alphabetical List of Options > CACHE_PARTITIONS Option*.

See *System Administration Guide: Volume 1 > Transactions and Versioning > Tools for Managing Locks*.

## -iqstart iqsrv15 Server Option

Provides startup diagnostics for dbspaces.

*Syntax*

**-iqstart** *N*

*Remarks*

The input parameter *N* is a number value that represents an integer bit mask. You may combine values to provide more than one feature. Output generated before the IQ message file is generated goes to the console. The **-z** startup switch provides additional startup and connection information.

The allowed values are as follows:

- N=1—returns basic information about the file names from `SYSIQFILES` that are used when opening the dbspace. It then displays the fully qualified names used. You can use this option to create a record of the files in use by the database in the IQ message file.
- N=2—stops after the transaction log replay before executing RecoveryComplete allowing you to examine the database without opening it all the way. You can combine **N=2** with other options. In certain modes using N=2 may rewrite the commit_identity, but does not otherwise modify the database in a permanent manner—the checkpoint that would commit the recovery actions is not allowed to complete. All recovery actions reexecute the next time the database is opened.
- N=4—returns full diagnostic information, including all rows of `SYSIQFILE`, the subset of file names selected if the database is a multiplex database, the fully resolved file names, each individual dbspace file header block, the database_identity, the commit_identity, each checkpoint log entry, and each transaction log entry.
- N=8—allows the file paths in `SYSIQFILE` to be overridden. Instead of the `SYSIQFILE` values, the file names `iqmsg.iqmsg, iqmain_1, iqmain_2, ..., iqtemp_1, iqtemp_2,` and so on. will be used. These may be links and must be in the same directory as the `.db` file. You may use a link to the actual `.db` file but if the server is given a link to a `.db` file that uses a transaction log relative to the database, the server looks for the transaction log relative to the link, rather than to the database. In this case, create a link for the transaction log also

## -iqtc iqsrv15 Server Option

Specifies IQ temporary store cache size, in MB.

*Syntax*

**-iqtc** *size*

*Remarks*

When using this parameter, always specify the value for the size, but no units of measurement; for example specify `32` instead of `32MB`. If you specify the unit of measurement, **start_iq** ignores this switch, unlike SQL Anywhere, which requires a unit of measurement.

Overrides default of 64MB. Applies to all databases started from the time the IQ server is started until the IQ server is shut down. In other words, if you start one database at server startup and another later, you need 2 * **-iqtc** available for the temp cache. In general, Sybase recommends that you do not run multiple databases with a Sybase IQ server.

Ensure that the IQ_SYSTEM_TEMP dbspace is at least as large as **-iqtc**.

## -iqtss iqsrv15 Server Option

Specifies the stack size, in KB, for server execution threads running either in the background or as part of a thread team assisting the main server connection thread.

*Syntax*

**-iqtss** *size*

*Remarks*

The default is 512KB on 64-bit platforms, and 200KB on 32-bit platforms.

## -iqwmem iqsrv15 Server Option

Creates a pool of “wired” memory on HP and Sun UNIX systems.

*Syntax*

**-iqwmem** *size*

*Remarks*

This memory is locked down so it cannot be paged by the operating system. Specify the memory size, in MB. Use this switch only if you have enough memory to dedicate for this purpose. Otherwise, you may cause serious performance degradation.

## -k iqsrv15 server option

Controls the collection of Performance Monitor statistics.

*Syntax*

```
iqsrv15 -k ...
```

*Default*

Performance Monitor statistics are collected

*Applies to*

All operating systems and database servers.

*Remarks*

If you specify -k when you start the database server, then the database server does not collect Performance Monitor statistics. The -k option does not affect the collection of column statistics used by the query optimizer.

This option should only be used in situations where the database server is running on a multi-processor computer where it can be shown by testing to improve performance. For most workloads, the benefit will be negligible, so use of this option is not recommended. When you disable the performance counters, this information is not available for analyzing performance problems.

You can also change the setting for the collection of Performance Monitor statistics using the sa_server_option system procedure.

## -kl iqsrv15 server option

Specifies the file name of the Kerberos GSS-API library (or shared object on Unix) and enables Kerberos authenticated connections to the database server.

### *Syntax*

```
iqsrv15 -kl GSS-API-library-file ...
```

### *Applies to*

All operating systems except Windows Mobile.

This option has not been fully tested with servers in the cloud.

### *Remarks*

This option specifies the location and name of the Kerberos GSS-API. This option is only required if the Kerberos client uses a different file name for the Kerberos GSS-API library than the default, or if there are multiple GSS-API libraries installed on the computer running the database server. A Kerberos client must already be installed and configured, and SSPI cannot be used by the database server.

Specifying this option enables Kerberos authentication to the database server.

### *Example*

The following command starts a database server that uses the `libgssapi_krb5.so` shared object for Kerberos authentication.

```
iqsrv15 -kl libgssapi_krb5.so -n my_server_princ /opt/myapp/
kerberos.db
```

## -kr iqsrv15 server option (deprecated)

Specifies the realm of the Kerberos server principal and enables Kerberos authenticated connections to the database server.

**Note:** The use of the -kr option is deprecated. It is recommended that you use the -kp option to specify the Kerberos server principal. When specifying -kp the server principal must have been extracted to the Kerberos keytab file on the computer running the database server.

### *Syntax*

```
iqsrv15 -kr server-realm ...
```

*Applies to*

All operating systems except Windows Mobile.

This option has not been fully tested with servers in the cloud.

*Remarks*

This option specifies the realm of the Kerberos server principal. Normally, the principal used by the database server for Kerberos authentication is *server-name@default-realm*, where *default-realm* is the default realm configured for the Kerberos client. Use this option if you want the server principal to use a different realm than the default realm, in which case the server principal used is *server-name@server-realm*.

Specifying this option enables Kerberos authentication to the database server.

The -kr option cannot be specified if the -kp option is specified.

*Example*

The following command starts a database server that accepts Kerberos logins and uses the principal my_server_princ@MYREALM for authentication.

```
iqsrv15 -kr MYREALM -n my_server_princ C:\kerberos.db
```

## -krb iqsrv15 server option

Enables Kerberos-authenticated connections to the database server.

*Syntax*

```
iqsrv15 -krb ...
```

*Applies to*

All operating systems except Windows Mobile.

This option has not been fully tested with servers in the cloud.

*Remarks*

This option enables Kerberos authentication to the database server. You must specify one or more of the -krb, -kl, and -kr options for the database server to be able to authenticate clients using Kerberos.

Before you can use Kerberos authentication, a Kerberos client must already be installed and configured on both the client and database server computers. Additionally, the principal *server-name@REALM* must already exist in the Kerberos KDC, and the keytab for the principal *server-name@REALM* must already have been securely extracted to the keytab file on the database server computer. The database server will not start if the -krb option is specified, but this setup has not been performed.

**Note:** The database server name cannot contain any of the following characters: /, \, or @, and database server names with multibyte characters cannot be used with Kerberos.

The login_mode database option must be set to allow Kerberos logins, and Kerberos client principals must be mapped to database user IDs using the GRANT KERBEROS LOGIN statement.

*Example*

For a Kerberos principal for the database server named my_server_princ@MYREALM, the following command starts a database server named my_server_princ.

```
iqsrv15 -krb -n my_server_princ C:\kerberos.db
```

## -ks iqsrv15 server option

Disables the creation of shared memory that the Performance Monitor uses to collect counter values from the database server.

*Syntax*

```
iqsrv15 -ks 0 ...
```

*Applies to*

Windows

*Remarks*

When you specify this option, the Performance Monitor does not show any database server, database, or connection statistics for the current database server.

## -ksc iqsrv15 server option

Specifies the maximum number of connections that the Performance Monitor can monitor.

*Syntax*

```
iqsrv15 -ksc integer ...
```

*Default*

10

*Applies to*

Windows

## -ksd iqsrv15 server option

Specifies the maximum number of databases that the Performance Monitor can monitor.

*Syntax*

```
iqsrv15 -ksd integer ...
```

*Default*
2

*Applies to*
Windows

## -m iqsrv15 server option

Truncates the transaction log when a checkpoint is done.

*Syntax*

```
iqsrv15 -m ...
```

*Applies to*
All operating systems and database servers.

This option does not apply to servers in the cloud.

*Remarks*
This option truncates the transaction log when a checkpoint is done, either at shutdown or as a result of a checkpoint scheduled by the server.

**Warning!** When this option is selected, there is no protection against media failure on the device that contains the database files.

This option provides a way to automatically limit the growth of the transaction log. Checkpoint frequency is still controlled by the checkpoint_time and recovery_time options (which you can also set on the command line).

The -m option is useful for limiting the size of the transaction log in situations where high volume transactions requiring fast response times are being processed, and the contents of the transaction log aren't being relied upon for recovery or replication. The -m option provides an alternative to operating without a transaction log at all, in which case a checkpoint would be required following each COMMIT and performance would suffer as a result. When the -m option is specified, there is no protection against media failure on the device that contains the database files. Other alternatives for managing the transaction log (for example, using the BACKUP statement and events) should be considered before using the -m option.

To avoid database file fragmentation, it is recommended that where this option is used, the transaction log be placed on a separate device or partition from the database itself.

When this option is used, no operations can proceed while a checkpoint is in progress.

**Warning!** Do not use the -m option with databases that are being replicated or synchronized.

## -n iqsrv15 server option

Sets the name of the database server.

*Syntax*

```
iqsrv15 -n server-name database-filename ...
```

*Default*

The name of the first database file (with the path and extension removed) that is started on the database server.

*Applies to*

All operating systems and database servers.

This option does not apply to servers in the cloud.

*Remarks*

When a database server starts, it attempts to become the default database server on that computer. The first database server to start when there is no default server becomes the default database server. Shared memory connection attempts on that computer that do not explicitly specify a database server name connect to the default server.

**Note:** It is recommended that you use the -xd option for database servers being used by deployed applications, and that all clients explicitly specify the name of the database server to which they should connect by using the ServerName (Server) connection parameter. This ensures that the database connects to the correct database server when a computer is running multiple database servers.

There is no character set conversion performed on the server name. If the client character set and the database server character set are different, using extended characters in the server name can cause the server to not be found. If your clients and servers are running on different operating systems or locales, you should use 7-bit ASCII characters in the server name.

Database server names must be valid identifiers. Long database server names are truncated to different lengths depending on the protocol. Database server names cannot:

begin with white space, single quotes, or double quotes
end with white space
contain semicolons, forward slashes (/), or backslashes (\)
be longer than 250 bytes
contain spaces when they are running on Unix

**Note:** On Windows and Unix, version 9.0.2 and earlier clients cannot connect to version 10.0.0 and later database servers with names longer than the following lengths:

40 bytes for Windows shared memory

31 bytes for Unix shared memory

40 bytes for TCP/IP

---

The server name specifies the name to be used in the ServerName (Server) connection parameter of client application connection strings or profiles. With shared memory, unless -xd is specified, a default database server is used if a server name is not specified and there is at least one database server running on the computer.

Running multiple database servers with the same name is not recommended.

---

**Note:** There are two -n options. The -n option is positional. If it appears before any database file names, it is a server option and names the server. If it appears after a database file name, it is a database option and names the database.

For example, the following command names the database server SERV and the database DATA:

```
iqsrv15 -n SERV sales.db -n DATA
```

---

## Database Server Naming Restrictions

If you use the **-n** switch in **start_iq [server-options]**, certain naming restrictions apply.

No character set is conversion performed on the server name. If the client character set and the database server character set differ, using extended characters in the server name can cause the server to not be found. If clients and servers run on different operating systems or locales, use 7-bit ASCII characters in the server name.

Database server names must be valid identifiers. Long database server names are truncated to different lengths depending on the protocol. Database server names cannot:

- Begin with white space, single quotes, or double quotes
- End with white space
- Contain semicolons
- Exceed 128 bytes

---

**Note:** On Windows and UNIX, Sybase IQ 12.7 and earlier clients cannot connect to Sybase IQ 15.x database servers with names longer than the following lengths:

- 40 bytes for Windows shared memory
- 31 bytes for UNIX shared memory
- 40 bytes for TCP/IP

---

The server name specifies the name to be used on client application connection strings or profiles. Running multiple database servers with the same name is not recommended.

**See also**

- *start_iq Server Options* on page 115
- *AWE Cache Sizes per Operating System* on page 127
- *Starting a Database Server with an AWE Cache* on page 129

## -o iqsrv15 server option

Prints all database server messages to the database server message log file.

*Syntax*

```
iqsrv15 -o filename ...
```

*Applies to*

All operating systems and database servers.

This option does not apply to servers in the cloud.

*Remarks*

Print all database server messages, including informational messages, errors, warnings, and MESSAGE statement output, to the specified file, and to the database server messages window. If you specify the -qi option with -o, all messages appear only in the database server message log file.

It is recommended that you do not end the file name with `.log` because this can create problems for utilities that perform operations using the transaction log.

To find the name of the database server message log file, run the following statement:

```
SELECT PROPERTY ( 'ConsoleLogFile' );
```

### start_iq Error Reporting

If there is a problem starting the server, **start_iq** returns a non zero value.

If you did not specify a log file after the **-o** switch on startup, the error is written to the first one of the following that is defined:

- (Windows) %ALLUSERSPROFILE%\SybaseIQ\IQ15_4\logfiles\
- (UNIX / Linux) $IQDIR15/logfiles/

## -oe iqsrv15 server option

Specifies a file name to log startup errors, fatal errors, and assertions.

*Syntax*

```
iqsrv15 -oe filename ...
```

*Applies to*

All operating systems and database servers.

### *Remarks*

Each line in the output log file is prefixed with the date and time. Startup errors include such errors as:

- Couldn't open/read database file: *database file*
- A database server with that name has already started

Fatal errors and assertions are logged to the Windows Application Event Log (except on Windows Mobile) or the Unix system log regardless of whether -oe is specified.

It is recommended that you do not end the file name with `.log` because this can create problems for utilities that perform operations using the transaction log.

## -on iqsrv15 server option

Specifies a maximum size for the database server message log, after which the file is renamed with the extension `.old` and a new file is started.

### *Syntax*

```
iqsrv15 -on { size[ k | m | g ] } ...
```

### *Applies to*

All operating systems and database servers.

This option does not apply to servers in the cloud.

### *Remarks*

The *size* is the maximum file size for the database server message log, in bytes. Use `k`, `m`, or `g` to specify units of kilobytes, megabytes, or gigabytes respectively. The minimum size limit is 10 KB. By default, there is no maximum size limit.

When the database server message log reaches the specified size, the database server renames the file with the extension `.old`, and starts a new file with the original name.

**Note:** If the `.old` database server message log file already exists, it is overwritten. To avoid losing old database server message log files, use the -os option instead.

This option cannot be used with the -os option.

It is recommended that you do not end the database server message log file name with `.log` because this can create problems for utilities that perform operations using the transaction log.

## -os iqsrv15 server option

Specifies a maximum size for the database server message log file, at which point the file is renamed.

### *Syntax*

```
iqsrv15 -os { size[ k | m | g ] } ...
```

### *Applies to*

All operating systems and database servers.

This option does not apply to servers in the cloud.

### *Remarks*

The *size* is the maximum file size for logging database server messages, in bytes. Use `k`, `m`, or `g` to specify units of kilobytes, megabytes, or gigabytes respectively. The minimum size limit is 10 KB. By default, there is no maximum size limit.

Before the database server logs output messages to the database server message log file, it checks the current file size. If the log message will make the file size exceed the specified size, the database server renames the database server message log file to `yymmddxx.slg`, where *yymmdd* represents the year, month, and day the file was created, and *xx* is a number that starts at 00 and continues incrementing.

This option allows you to identify old database server message log files that can be deleted to free up disk space.

This option cannot be used with the -on option.

It is recommended that you do not end the database server message log file name with `.log` because this can create problems for utilities that perform operations using the transaction log.

## -ot iqsrv15 server option

Truncates the database server message log file and appends output messages to it.

### *Syntax*

```
iqsrv15 -ot logfile ...
```

### *Applies to*

All operating systems and database servers.

*Remarks*

The functionality is the same as the -o option except the database server message log file is truncated before any messages are written to it. To find the name of the database server message log file, execute the following statement:

```
SELECT PROPERTY ( 'ConsoleLogFile' );
```

It is recommended that you do not end the database server message log file name with `.log` because this can create problems for utilities that perform operations using the transaction log.

## -p iqsrv15 server option

Sets the maximum size of communication packets.

*Syntax*

```
iqsrv15 -p integer ...
```

*Default*

7300 bytes (all operating systems except Windows Mobile)

1460 bytes (Windows Mobile)

*Applies to*

All operating systems and database servers.

*Remarks*

The minimum value is 500 bytes and the maximum value is 16000.

You can change the communication buffer size for a connection by setting the CommBufferSize (CBSIZE) connection parameter.

## -pc iqsrv15 server option

Compresses all connections except for same-computer connections.

*Syntax*

```
iqsrv15 -pc ...
```

*Applies to*

All operating systems and network servers, except web servers.

*Remarks*

The packets sent between a client and database server can be compressed using the -pc option. Compressing a connection may improve performance under some circumstances. Large data transfers with highly compressible data tend to get the best compression rates. This option can be overridden for a particular client by specifying COMPRESS=NO in the client's connection parameters.

By default, connections are not compressed. Specifying the -pc option compresses all connections except same-computer connections, web services connections, and TDS connections. TDS connections (including jConnect) do not support communication compression.

Same-computer connections over any communication link are not compressed, even if the -pc option or COMPRESS=YES connection parameter is used.

## -pt iqsrv15 server option

Increases or decreases the size limit at which packets are compressed.

*Syntax*

```
iqsrv15 -pt size ...
```

*Default*

120 bytes

*Applies to*

All operating systems and network servers.

*Remarks*

This parameter takes an integer value representing the minimum byte-size of packets to be compressed. Values less than 80 are not recommended.

Under some circumstances, changing the compression threshold can help performance of a compressed connection by allowing you to compress packets only when compression will increase the speed at which the packets are transferred. The default setting should be appropriate for most cases.

If both client and server specify different compression threshold settings, the client setting applies.

## -qi iqsrv15 server option

Controls whether database server system tray icon and database server messages window appear.

*Syntax*

```
iqsrv15 -qi ...
```

*Applies to*

Windows

*Remarks*
This option leaves no visual indication that the server is running, other than possible startup error windows. You can use either (or both) the -o or -oe log files to diagnose errors.

## -qp iqsrv15 server option

Specifies that messages about performance do not appear in the database server messages window.

*Syntax*

```
iqsrv15 -qp ...
```

*Applies to*
All operating systems and database servers.

*Remarks*
Do not display messages about performance in the database server messages window. Messages that are suppressed include the following:

- No unique index or primary key for table '*table-name*'
- Database file "*mydatabase.db*" consists of *nnn* fragments

## -qs iqsrv15 server option

Suppresses startup error windows.

*Syntax*

```
iqsrv15 -qs ...
```

*Applies to*
Windows

*Remarks*
This option suppresses startup error windows. Examples of startup errors include the database server not being able to open or read a database file or a database server not starting because another database server with the specified name is already running.

On Windows platforms, if the server isn't being started automatically, these errors appear in a window and must be cleared before the server stops. These windows do not appear if the -qs option is used.

If there is an error loading the language DLL, no window appears if -qs was specified on the command line and not in @data. This error isn't logged to the -o or -oe logs, but rather to the Windows Application Event Log (except on Windows Mobile).

Usage errors are suppressed if -qs is on the command line, but not in @data expansion.

## -qw iqsrv15 server option

Specifies that the database server messages window does not appear.

*Syntax*

```
iqsrv15 -qw ...
```

*Applies to*

All operating systems and database servers.

*Remarks*

This option suppresses the database server messages window. On Windows platforms, the database server system tray icon is still visible. You can use either (or both) the -o or -oe log files to diagnose errors.

## -s iqsrv15 server option

Sets the user ID for Syslog messages.

### Applies to

Unix

### Remarks

Sets the system user ID used in messages to the Syslog facility. The default is user for database servers that are started in the foreground, and daemon for those that are run in the background (for example, started by dbspawn, started automatically by a client, or started with the -ud database server option).

A value of none prevents any Syslog messages from being logged. The local*n* argument allows you to use a facility identifier to redirect messages to a file. You can specify a number between 0 and 7, inclusive, for *n*. Refer to the Unix Syslog(3) man page for more information.

*Redirect messages to a file using a facility identifier*

The following steps illustrate how to redirect messages on Solaris, but you can also do this on Linux, and IBM AIX. On other platforms, such as HP-UX, the `syslog.conf` file is found in a different location. You can place the `/var/adm/sqlanywhere` file in whatever location you want.

1. Choose a unique facility identifier that isn't already being used by another application that is running on your system.

   You can do this by looking in the `/etc/syslog.conf` file to see of any of the local*n* facilities are referenced.
2. Edit the `/etc/syslog.conf` file and add the following line, where local*n* is the facility identifier you chose in step 1:

```
localn.err;localn.info;localn.notice  /var/adm/sqlanywhere
```

3. Create the `/var/adm/sqlanywhere` file:

   ```
   touch /var/adm/sqlanywhere
   ```

4. Tell the syslogd process that you have modified the `syslog.conf` file by finding the process ID of syslogd:

   ```
   ps -ef | grip syslogd
   ```

   and then running the following command where *pid* is the process ID of syslogd:

   ```
   kill -HEAP pid
   ```

5. Start your database server with the following command, where `localn` is the facility identifier you chose in step 1:

   ```
   iqsrv15 -s localn ...
   ```

   Now any messages that the database server reports to Syslog are redirected to the `/var/adm/sqlanywhere` file.

## -sb iqsrv15 server option

Specifies how the database server reacts to broadcasts.

### *Applies to*

TCP/IP.

This option has not been fully tested with servers in the cloud.

### *Remarks*

By default, the database server starts one or more UDP listeners so that it can receive UDP broadcasts from client applications and respond to these broadcasts appropriately.

For embedded SQL connections the dblocate utility and db_locate_servers function cause broadcast packets to be sent out on local networks in an attempt to find all database servers. The UDP listener within the database server then responds back to the sender with information about how to connect to the database server.

The UDP listener can also be involved in client connection requests.

The -sb option controls the behavior of the UDP listeners within the database server.

Using -sb 0 stops the database server from starting any UDP broadcast listeners. This behavior forces clients to use a Host connection parameter or Host protocol option when connecting to the database server. In addition if the server is using a port other than the default port (2638), then the client must also specify the server's port. When -sb 0 is specified, the database server cannot respond to broadcast requests from dblocate, and therefore the database server is not included in the dblocate output.

Using -sb 1 stops the database server from responding to broadcasts from dblocate, so it is not found or listed. The UDP listeners still start so that they can respond to requests from clients that are attempting to connect to the database server.

## -sf iqsrv15 server option

Enables and disables features for databases running on the current database server.

### *Syntax*

```
iqsrv15 -sf feature-list ...
```

### *Allowed values*

The following *feature-name* values are supported:

- **none –** Specifies that no features are disabled.
- **all –** Disables all features that can be disabled including the following groups.
    - **client –** Disables all features that allow access to client-related input/output. This includes access to the client computing environment. This set consists of the following features.
        - **read_client_file –** Disables the use of statements that can cause a client file to be read. For example, the READ_CLIENT_FILE function and the LOAD TABLE statement.
        - **write_client_file –** Disables the use of all statements that can cause a client file to be written to. For example, the UNLOAD statement and the WRITE_CLIENT_FILE function.
    - **local –** Disables all local-related features. This includes access to the server computing environment. This set consists of the `local_call`, `local_db`, `local_io`, and `local_log` feature subsets described below.
        - **local_call –** Disables all features that provide the ability to execute code that is not directly part of the server and is not controlled by the server. This set consists of the following features.
            - **cmdshell –** Disables the use of the xp_cmdshell procedure.
            - **external_procedure –** Disables the use of external stored procedures. This setting does not disable the use of the xp_* system procedures (such as xp_cmdshell, xp_readfile, and so on) that are built into the database server. Separate feature control options are provided for these system procedures.
            - **external_procedure_v3 –** See the User-Defined Functions guide.
            - **java –** Disables the use of Java-related features, such as Java procedures.
        - **local_db –** Disables all features related to database files. This set consists of the following features.

- **backup –** Disables the use of the BACKUP statement, and therefore, the ability to run server-side backups. You can still perform client-side backups using dbbackup.
- **restore –** Disables the use of the RESTORE DATABASE statement.
- **database –** Disables the use of the CREATE DATABASE, ALTER DATABASE, DROP DATABASE, CREATE ENCRYPTED FILE, CREATE DECRYPTED FILE, CREATE ENCRYPTED DATABASE, and CREATE DECRYPTED DATABASE statements.
- **dbspace –** Disables the use of the CREATE DBSPACE, ALTER DBSPACE, and DROP DBSPACE statements.

- **local_io –** Disables all features that allow direct access to files and their contents. This set consists of the following features.

  - **read_file –** Disables the use of statements that can cause a local file to be read. For example, the xp_read_file system procedure, the LOAD TABLE statement, and the use of OPENSTRING( FILE ... ). The alternate names load_table and xp_read_file are deprecated.
  - **write_file –** Disables the use of all statements that can cause a local file to be written to. For example, the UNLOAD statement and the xp_write_file system procedure. The alternate names unload_table and xp_write_file are deprecated.
  - **delete_file –** Disables the use of all statements that can cause a local file to be deleted. For example, it disables the use of the db_delete_file DBLib function, which deletes database files. The db_delete_file function is used by the dbbackup -x and -xo options, so securing db_delete_file causes dbbackup to fail if the -x or -xo options are specified.
  - **directory –** Disables the use of directory class proxy tables. This feature is also disabled when remote_data_access is disabled.

- **local_log –** Disables all logging features that result in creating or writing data directly to a file on disk. This set consists of the following features.

  - **request_log –** Disables the ability to change the request log file name and also disables the ability to increase the limits of the request log file size or number of files. You can specify the request log file and limits on this file, in the command to start the database server; however, they cannot be changed once the server is started. When request log features are disabled, you can still turn request logging on and off, and reduce the maximum file size and number of request logging files.
  - **console_log –** Disables the ability to change the database server message log file name using the ConsoleLogFile option of the sa_server_option system procedure. It also disables the ability to increase the maximum size of the log file using the ConsoleLogMaxSize option of the sa_server_option system procedure. You can specify a server log file and its size when starting the database server.

    - **webclient_log –** Disables the ability to change the web service client log file name using the WebClientLogFile option of the sa_server_option system procedure. You can specify a web service client log file when starting the database server.
- **remote –** Disables all features that allow remote access or communication with remote processes. This set consists of the following features.
    - **remote_data_access –** Disables the use of any remote data access services, such as proxy tables.
    - **send_udp –** Disables the ability to send UDP packets to a specified address using the sa_send_udp system procedure.
    - **web_service_client –** Disables the use of web service client stored procedure calls (that is, stored procedures that issue HTTP requests).

### *Applies to*

All operating systems and database servers.

This option does not apply to servers in the cloud.

### *Remarks*

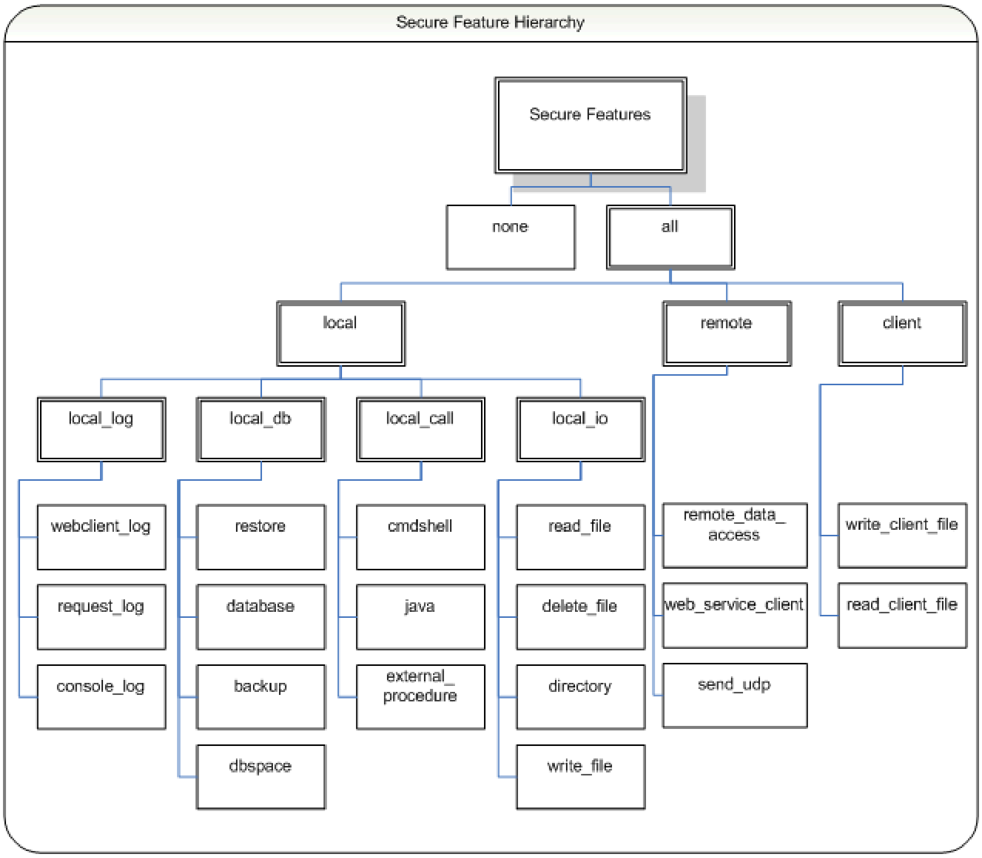
This option allows you to enable and disable features for a database server. These settings affect all databases running on the database server. You can enable all disabled (secured) features for a connection by setting the secure_feature_key option to the key specified by the -sk option. Any connection that sets the secure_feature_key option to the key specified by -sk can also change the set of secured features for a database server using the `SecureFeatures` property of the sa_server_option system procedure.

The *feature-list* is a comma-separated list of feature names or feature sets to secure for the database server. Use *feature-name* to indicate that the feature should be disabled, and `-`*feature-name* to indicate that the feature should be removed from the disabled features list. For example, the following command indicates that only dbspace features are enabled:

```
iqsrv15 -n secure_server -sf all,-dbspace
```

### *Feature set hierarchy*

The following diagram lists all the feature set keywords and their hierarchy. For example, `local_io` encompasses the `read_file`, `write_file`, `delete_file`, and `directory` features.



*Example*

The following command starts a database server named secure_server with access to the request log and with all remote data access features disabled. The key specified by the -sk option can be used later with the secure_feature_key database option to enable these features for a specific connection.

```
iqsrv15 -n secure_server -sf request_log,remote -sk j978kls12
```

If a user connected to a database running on the secure_server database server sets the secure_feature_key option to the value specified by -sk, that connection has access to the request log and remote data access features:

```
SET TEMPORARY OPTION secure_feature_key = 'j978kls12';
```

The following command disables all features, with the exception of local database features:

```
iqsrv15 -n secure_server -sf all,-local_db
```

## -sk iqsrv15 server option

Specifies a key that can be used to enable features that are disabled for the database server.

*Syntax*

```
iqsrv15 -sk key ...
```

*Applies to*

All operating systems and database servers.

This option does not apply to servers in the cloud.

*Remarks*

When you secure features for a database server using the -sf option, you can also include the -sk option, which specifies a key that can be used with the secure_feature_key database option to enable secured features for a connection. That connection can also use the sa_server_option system procedure to modify the features or feature sets that are secured for all databases running on the database server.

If the secure_feature_key option is set to any value other than the one specified by -sk, no error is given, and the features specified by -sf remain secured for the connection.

*Example*

The following command starts a database server named secure_server with access to the backup features disabled. The key specified by the -sk option can be used later to enable these features for a specific connection.

```
iqsrv15 -n secure_server -sf backup -sk j978kls12
```

Setting the secure_feature_key option to the value specified by -sk for a connection to a database running on the secure_server database server allows that connection to perform backups or change the features that are disabled on the secure_server database server:

```
SET TEMPORARY OPTION secure_feature_key = 'j978kls12';
```

The user could then disable the use of all secured features for databases running on secure_server by executing the following command:

```
CALL sa_server_option( 'SecureFeatures', 'all' );
```

## -su iqsrv15 server option

Sets the password for the DBA user of the utility database (utility_db), or disable connections to the utility database.

*Syntax*

```
iqsrv15 -su password ...
```

*Applies to*

All operating systems and database servers.

This option does not apply to servers in the cloud.

*Remarks*

This option specifies the initial password for the DBA user of the utility database. The password is case sensitive. You can specify `none` for the password to disable all connections to the utility database. To avoid having the utility database password in clear text on the command line, you can use dbfhide to obfuscate a file containing the password, and then reference the obfuscated file on the command line.

- If you are using a personal database server and do not specify the -su option, connections to the utility database are allowed with the DBA user ID and any password.
- If you are using the network database server and do not specify the -su option, connections to the utility database are not allowed unless the `util_db.ini` file exists and the user ID is DBA with a password that matches the password in the `util_db.ini` file. On a network server, if both -su and `util_db.ini` are used, `util_db.ini` is ignored. The `util_db.ini` file is deprecated.

  Specifying a password for the utility database for a network database server is useful for the cases when the database server is running, but it is not possible to connect to the database.

You can execute a CREATE USER DBA IDENTIFIED BY *new-password* statement while connected to utility_db to change the password for the DBA user of the utility database. The REVOKE CONNECT FROM DBA statement can be used to disable connections to the utility_db database.

*Example*

The following command disables all connections to the utility database:

```
iqsrv15 -su none c:\inventory.db
```

In the following example, the file named `util_db_pwd.cfg` that contains the utility database password is obfuscated using dbfhide and renamed `util_db_pwd_hide.cfg`:

```
dbfhide util_db_pwd.cfg util_db_pwd_hide.cfg
```

The `util_db_pwd_hide.cfg` file can then be used to specify the utility database password:

```
iqsrv15 -su @util_db_pwd_hide.cfg -n my_server c:\inventory.db
```

## -ti iqsrv15 server option

Disconnects inactive connections.

*Syntax*

```
iqsrv15 -ti minutes ...
```

*Default*
240 (4 hours)

*Applies to*
All operating systems and database servers.

This option has not been fully tested with servers in the cloud.

*Remarks*
Disconnects connections that haven't submitted a request for the specified number of *minutes*. The maximum value is 32767. A client computer in the middle of a database transaction holds locks until the transaction is ended or the connection is disconnected. The -ti option is provided to disconnect inactive connections, freeing their locks.

The -ti option is very useful when used in conjunction with iqsrv15 since most connections will be over network links (TCP).

The -ti option is useful only for local TCP/IP connections. Using -ti has no effect on connections to a local server using shared memory, but you can specify a timeout for the connection that applies to shared memory connections. By default, the personal database server does not start TCP/IP.

Setting the value to zero disables checking of inactive connections, so that no connections are disconnected. If the Idle connection parameter is not used, then the idle timeout value for TCP/IP connections is controlled by the -ti database server option. If both the -ti database server option and the Idle connection parameter are specified, then the idle timeout value is controlled by the connection parameter.

## -tl iqsrv15 server option

Sets the period at which to send liveness packets.

*Syntax*

```
iqsrv15 -tl seconds ...
```

*Applies to*
All database servers using TCP/IP.

This option has not been fully tested with servers in the cloud.

*Remarks*
A liveness packet is sent periodically across a client/server TCP/IP communications protocol to confirm that a connection is intact. If the server runs for a LivenessTimeout period (default 2 minutes) without detecting a liveness packet on a connection, the communication is severed, and the server drops the connection associated with that client. Unix non-threaded clients and TDS connections do not do liveness checking.

The -tl option on the server sets the LivenessTimeout value for all clients that do not specify a liveness period.

Liveness packets are sent when a connection hasn't sent any packets for between one third and two thirds of the LivenessTimeout value.

When there are more than 200 connections, the server automatically calculates a higher LivenessTimeout value based on the stated LivenessTimeout value, so the server can handle a large number of connections more efficiently. Liveness packets are sent between one third and two thirds of the LivenessTimeout on each idle connection. Large numbers of liveness packets aren't sent at the same time. Liveness packets that take a long time to send could be sent after two thirds of the LivenessTimeout. A warning appears in the database server message log if the liveness sends take a long time. If this warning occurs, consider increasing the LivenessTimeout value.

Although it isn't generally recommended, you can disable liveness by specifying the following:

```
iqsrv15 -tl 0 -n my_server
```

Rather than disabling the LivenessTimeout option, consider increasing the value to one hour as follows:

```
iqsrv15 -tl 3600 -n my_server
```

## -tmf iqsrv15 server option

Forces transaction manager recovery for distributed transactions.

### *Syntax*

```
iqsrv15 -tmf ...
```

### *Applies to*

Windows.

This option does not apply to servers in the cloud.

### *Remarks*

Used during recovery of distributed transactions when the distributed transaction coordinator isn't available. If DTC cannot be located, the outstanding operations are rolled back and recovery continues. It can also be used when starting a database with distributed transactions in the transaction log on a platform where the DTC isn't available.

**Warning!** If you use this option, distributed transactions are not recovered properly. It is not intended for routine use.

## -tq iqsrv15 server option

Shuts down the server at a specified time.

### *Syntax*

```
iqsrv15 -tq { datetime | time } ...
```

### *Applies to*

All operating systems and database servers.

This option does not apply to servers in the cloud.

### *Remarks*

This option is useful for setting up automatic off-line backup procedures.

The format for the time is in *hh*:*mm* (24 hour clock), and can be preceded by an optional date. If a date is specified, the date and time must be enclosed in double quotes and be in the format *YYYY/MM/DD HH*:*MM*.

## -u iqsrv15 server option

Opens files using the operating system disk cache.

### *Syntax*

```
iqsrv15 -u ...
```

### *Applies to*

Windows, Unix.

This option has not been fully tested with servers in the cloud.

### *Remarks*

Files are opened using the operating system disk cache in addition to the database cache.

While the operating system disk cache may improve performance, better performance is often obtained by using only the database cache.

If the server is running on a dedicated computer, you shouldn't use the -u option, as the database cache itself is generally more efficient. You may want to use the -u option if the server is running on a computer with several other applications (so that a large database cache may interfere with other applications) and yet IO-intensive tasks are run intermittently on the server (so that a large cache will improve performance).

## -uf iqsrv15 server option

Specifies the action to take when a fatal error occurs.

*Syntax*

```
iqsrv15 -uf action ...
```

*Default*

default

*Allowed values*

- **abort –** the Unix abort function is called, and a core file is generated.
- **default –** the database server always behaves in the same manner as abort, except when a device-full fatal error occurs. In this case, it behaves in the same manner as defunct. This action prevents the system from trying to write a core file on a full device. This is the default behavior.
- **defunct –** the database server continues running and does not call abort. Any new connection attempts made to the database server receive the SQL error of the original fatal error.

*Applies to*

Unix

## -z iqsrv15 server option

Displays diagnostic communication messages, and other messages, for troubleshooting purposes.

*Syntax*

```
iqsrv15 -z ...
```

*Applies to*

All operating systems and database servers.

This option does not apply to servers in the cloud.

*Remarks*

This should only be used when tracking problems. The information appears in the database server messages window.

## -ze iqsrv15 server option

Displays database server environment variables in the database server messages window.

*Syntax*

```
iqsrv15 -ze ...
```

*Applies to*

All operating systems and database servers except Windows Mobile.

*Remarks*

When you specify the -ze option, environment variables are listed in the database server messages window on startup. You can log the contents of the database server messages window to a file by specifying the -o option when starting the database server.

*Example*

The following command starts a database server named myserver, and outputs the environment variables set for the server to the database server messages window and the file `server-log.txt`.

```
iqsrv15 -n myserver -ze -o server-log.txt
```

## -zl iqsrv15 server option

Turns on capturing of the most recently-prepared SQL statement for each connection to databases on the server.

*Syntax*

```
iqsrv15 -zl ...
```

*Applies to*

All operating systems and database servers.

This option does not apply to servers in the cloud.

*Remarks*

This feature can also be turned on using the RememberLastStatement server setting. You can obtain the most recently-prepared SQL statement for a connection using the LastStatement value of the CONNECTION_PROPERTY function. The sa_conn_activity stored procedure allows you to obtain the most recently-prepared SQL statement for all current connections to databases on the server.

The LastStatement value is set when a statement is prepared, and is cleared when a statement is dropped. Only one statement string is remembered for each connection.

If sa_conn_activity reports a non-empty value for a connection, it is most likely the statement that the connection is currently executing. If the statement had completed, it would likely have been dropped and the property value would have been cleared. If an application prepares multiple statements and retains their statement handles, the LastStatement value does not reflect what a connection is currently doing.

For stored procedure calls, only the outermost procedure call appears, not the statements within the procedure.

**Warning!** When -zl is specified or when the RememberLastStatement server setting is turned on, any user can call the sa_conn_activity system procedure or obtain the value of the LastStatement connection property to find out the most recently-prepared SQL statement for any other user. This option should be used with caution and turned off when it isn't required.

## -zn iqsrv15 server option

Specifies the number of request log file copies to retain.

### *Syntax*

```
iqsrv15 -zn integer
```

### *Applies to*

All operating systems and database servers.

This option does not apply to servers in the cloud.

### *Remarks*

If request logging is enabled over a long period of time, the request log file can become large. The -zn option allows you to specify the number of request log file copies to retain. It only takes effect if -zs is also specified. The -zs option allows you to create a new log file and rename the original log file when the original log file reaches a specified size.

For example, if you redirect request logging information to the file `req.out`, and specify five request log file copies using the -zn option, the server creates files in the following order: `req.out.1`, `req.out.2`, `req.out.3`, `req.out.4`, and `req.out.5`. When these files exist and the active request log fills again, the following happens:

- `req.out.1` is deleted
- the files `req.out.2` to `req.out.5` are renamed `req.out.1` to `req.out.4`
- the copy of the active log is renamed `req.out.5`

Request logging is turned on using the -zr option and redirected to a separate file using the -zo option. You can also set the number of request logs using the sa_server_option system procedure where *nn* specifies the number of request log file copies:

```
CALL sa_server_option('RequestLogNumFiles',nn);
```

*Example*

In the following example, entered all on one line, request logging information is output to a request log file named `mydatabase.log`, which has a maximum size of 10 KB, and three copies of the request log are kept:

```
iqsrv15 "c:\my data\mydatabase.db" -zr all -zn 3
 -zs 10 -zo mydatabase.log
```

## -zo iqsrv15 server option

Redirects request logging information to a file separate from the regular log file.

*Syntax*

```
iqsrv15 -zo filename ...
```

*Applies to*

All operating systems and database servers.

This option does not apply to servers in the cloud.

*Remarks*

Request logging is turned on using the -zr option. You can direct the output from this file to a different file that is not the regular log file by specifying the -zo option.

This option also prevents request logging from appearing in the database server messages window.

## -zoc iqsrv15 server option

Redirects web service client information to a file.

*Syntax*

```
iqsrv15 -zoc filename ...
```

*Applies to*

All operating systems and database servers.

This option has not been fully tested with servers in the cloud.

*Remarks*

The web service client log file contains HTTP requests and transport data recorded for outbound web service client calls. Logging is enabled automatically when you specify the -zoc server option. You can enable and disable logging to this file using the sa_server_option system procedure:

```
CALL sa_server_option( 'WebClientLogging', 'ON' );
```

*Example*

The following command starts the database server so that it listens for HTTP web requests on port 80, and logs outbound web service client information to the file `clientinfo.txt`:

```
iqsrv15 web.db -xs HTTP(PORT=80) -zoc clientinfo.txt
```

## -ud iqsrv15 Server Option

(UNIX servers) Causes the process to run as a daemon in the root directory. Sybase recommends that you do not use this switch in IQ servers.

## -ut iqsrv15 Server Option

(UNIX servers) Causes the server to touch catalog store temporary files at intervals specified by *min*.

*Syntax*

**-ut** *min*

## -v iqsrv15 Server Option

Displays the database server version in a message box (Windows) or in a version string (UNIX / Linux).

*Syntax*

**-v** or **-v2**

## -x iqsrv15 Server Option

Specifies server-side network communications protocols.

*Syntax*

**-x** *list*

*Remarks*

*list* is a comma-separated list of **tcpip** or **namedpipes** settings. For example:

```
-x tcpip,ipx
```

allows only TCP/IP and IPX communications.

The default is to try all settings supported by the database server on your operating system.

For some protocols, you can provide additional parameters, in this format:

```
-x tcpip(PARM1=value1;PARM2=value2;...)
```

For UNIX, quotation marks are required if more than one parameter is supplied:

```
-x "tcpip(PARM1=value1;PARM2=value2;...)"
```

For information on available communication protocols, see *System Administration Guide: Volume 1 > Connection and Communication Parameters*.

See *SQL Anywhere Server – Database Administration > Starting and Connecting to Your Database > The database server > Database server options > -x dbeng12/dbsrv12 server option*.

**Note:** This reference points to SQL Anywhere documentation.

## -xd iqsrv15 Server Option

Prevents the database server from becoming the default database server.

When a database server starts, it attempts to become the default database server on that computer. The first database server to start when there is no default server becomes the default database server. Shared memory connection attempts on that computer that do not explicitly specify a database server name connect to the default server.

Specifying this option prevents the database server from becoming the default database server. If this option is specified, clients that do not specify a database server name cannot find the database server over shared memory. The **-xd** option also prevents the database server from using the default TCP port. If a TCP port is not specified, the database server uses a port other than port 2638.

## -xs iqsrv15 Server Option

Specifies server-side Web services communications protocols.

```
-xs {protocol, ... }
protocol:
{NONE
| HTTP [(option=value;...)]
| HTTPS [(option=value;...)]
HTTPS-only options:
FIPS={Y | N}
IDENTITY=server-identity-filename
IDENTITY_PASSWORD=password
```

Use the **-xs** option to specify the Web protocols you want to use to listen for client connection broadcasts.

If you do not specify the **-xs** option, the server does not attempt to listen for Web requests.

If you specify one or more protocols, the server attempts to listen for client requests using the specified protocol(s).

You can use the HTTPS or the FIPS-approved HTTPS protocols for transport-layer security.

Regardless of the settings you choose for the **-xs** option, the server always listens for connection broadcasts using the shared memory protocol. You can specify any of the following:

- option—Use the supported network protocol option for a protocol.
- HTTP—Listen for web requests by the client using the HTTP protocol. The default port on which to listen is 80.
- HTTPS—Listen for web requests by the client using the HTTPS protocol. The default port on which to listen is 443. You must specify the server's certificate and password to use HTTPS. The password must be an RSA certificate because HTTPS uses RSA encryption.

  The SQL Anywhere HTTP server supports HTTPS connections using SSL version 3.0 and TLS version 1.0.

  You can specify HTTPS, or HTTPS with FIPS=Y for FIPS-approved RSA encryption. FIPS-approved HTTPS uses a separate approved library, but is compatible with HTTPS.

  - server-identity-filename–the path and file name of the server identity. For HTTPS, you must use an RSA certificate.
  - password –the password for the server private key. You specify this password when you create the server certificate.
- NONE—do not listen for web requests. This is the default.

For UNIX, you must use quotation marks if you are supplying more than one parameter:

```
-xs "http(OPTION1=value1;OPTION2=value2;...)"
```

This command allows only shared memory and TCP/IP communications:

```
start_iq web.db -xs http(port=80)
```

See *System Administration Guide: Volume 1 > Connection and Communication Parameters > Network communications parameters and CommLinks connection parameter [Links]*.

See *SQL Anywhere Server – Database Administration > Starting and Connecting to Your Database > The database server > Database server options > --xs dbeng12/dbsrv12 server option*.

**Note:** This reference points to SQL Anywhere documentation.

## -zr iqsrv15 Server Option

Enables request-level logging of operations.

### *Syntax*

**-zr** *level*

### *Remarks*

Specify maximum size of file for server request logging.

- All—logs all SQL statements and other requests to the server.
- None—turns off SQL statement logging. This is the default.
- SQL—logs the following types of requests only:
  - CONTROL_START_DATABASE

- CONTROL_STOP_ENGINE
- CONTROL_STOP_DATABASE
- STMT_PREPARE
- STMT_EXECUTE
- STMT_EXECUTE_IMM
- STMT_EXECUTE_ANY_IMM
- SQL_OPTION_SET
- BACKUP
- DELETE_FILE
- COMMIT
- ROLLBACK
- PREPARE_TO_COMMIT
- CONNECT
- DISCONNECT
- BEGIN_TRANSACTION
- STMT_DROP
- CURSOR_OPEN
- CURSOR_EXPLAIN
- CURSOR_CLOSE
- CURSOR_RESUME
- Errors

**-zr** also prevents request-level logging from appearing in the console. See *System Administration Guide: Volume 1 > Troubleshooting Hints > Diagnostic Tools> Logging Server Requests*.

For information about reading the **-zr** log output, see *System Administration Guide: Volume 1 > Troubleshooting Hints > Diagnostic Tools > Logging Server Requests > Request Log File Analysis*.

See also *SQL Anywhere Server - SQL Usage > Monitoring and Improving Database Performance > Improving database performance > Other diagnostic tools and techniques > Request logging*.

**Note:** This reference points to SQL Anywhere documentation.

See also **-zo file** and **-zs { integer | integerG | integerK | integerM } …** command-line switches.

## -zs iqsrv15 Server Option

Limits the size of the request-level logging file.

*Syntax*

**-zs {size [ k | m | g ] }**

*Remarks*

Turn on request-level logging using the **-zr** switch, and redirect to a separate file using the **-zo** switch. You limit the size of the file using the **-zs** switch.

Size is the maximum file size for the request log, in bytes. Use **k**, **m**, or **g** to specify units of kilobytes, megabytes, or gigabytes, respectively.

When the request log file reaches the size specified by either the **-zs** option or the **sa_server_option** system procedure, the file is renamed with the extension `.old` appended (replacing an existing file with the same name if one exists). The request-level log file is then restarted.

If you specify **-zs 0**, then there is no maximum size for the request logging file, and the file is never renamed. This is the default value.

**Note:** If the size of the query text being written to request log exceeds the specified limit, the query text is not truncated and is logged in its entirety.

The following example shows how the **-zs** option is used to control log file size. Suppose you start a database server with the following options on the command line:

```
-zr all -zs 10 -zo mydatabase.log
```

A new log file `mydatabase.log` is created. When this file reaches 10K in size, any existing `mydatabase.old` files are deleted, `mydatabase.log` is renamed to `mydatabase.old`, and a new `mydatabase.log` file is started. This process is repeated each time the `mydatabase.log` file reaches 10K.

See *SQL Anywhere Server – Database Administration > Starting and Connecting to Your Database > The database server > Database server options > -zs dbeng12/dbsrv12 server option*.

**Note:** This reference points to SQL Anywhere documentation.

## start_iq Database File Parameters

This topic lists the parameters of the database server/database file. You specify the database file after the server options in the command syntax.

**Table 32. start_iq database file parameters**

| Parameter | Description |
|---|---|
| **-n** *server-name* | Specifies the name of the database server. |

| Parameter | Description |
|---|---|
| **database-file** | Specifies the database file name. If *database-file* is specified without a file extension, Sybase IQ looks for *database-file* with extension `.db`.<br><br>If you use a relative path, the path is read relative to the current working directory of the server. You can supply a full path.<br><br>On Windows you can supply a path that conforms to the Universal Naming Convention (UNC) format:<br><br>`\\server\volume\path\file.ext` |

**Warning!** The database file must be on the same machine as the database server. Managing a database file that is located on a network drive can lead to file corruption.

## start_iq Database Options

Available switches for the **start_iq** *database-options* parameters.

Specify these options after the database file. These options apply only to that database. These options apply only to the preceding database in the command syntax.

There are two forms of syntax for **start_iq** database options:

- When specifying options in a configuration file, do not enclose the option value in quotation marks. For example:
  ```
  iqdemo.db -ek xxx
  ```
- When specifying options on the command line, enclose the option value in quotation marks. For example:
  ```
  start_iq @iqdemo.cfg iqdemo.db -ek 'xxx'
  ```

**See also**

- *dblocate Database Administration Utility* on page 67

### -a iqsrv15 database option

Applies the named transaction log.

*Syntax*

```
iqsrv15 [ server-options ] database-file -a log-filename ...
```

*Applies to*

All operating systems and database servers.

*Remarks*

This option is used to recover from media failure on the database file. When this option is specified, the database server applies the log and then shuts down—it doesn't continue to run. If you need to apply multiple transaction logs, you must know the correct order in which to apply them when using -a. The database server automatically applies multiple transaction logs in the correct order if you use the -ad or -ar option.

The -a database option must be specified after the *database-file*, and applies only to that database.

Specifying a cache size when starting the server can reduce recovery time.

*Example*

The following example, entered all on one line, applies the log file `iqdemo.log` to a backup copy of the sample database.

```
iqsrv15 "c:\backup\iqdemo.db" -a "c:\backup\iqdemo.log"
```

## -ad iqsrv15 database option

Specifies the directory containing transaction log files to be applied to the database.

*Syntax*

```
iqsrv15 [ server-options ] database-file -ad log-directory ...
```

*Applies to*

All operating systems and database servers.

*Remarks*

When you include the -ad option, the specified directory is scanned for transaction log files associated with the database. Transaction log files with starting log offsets greater than or equal to the start log offset stored in the database file are applied, in log offset order. Once all the transaction log files have been applied, the database is stopped. You must also specify the -as option if you want the database to continue running once the transaction log files have been applied.

If the specified log directory uses a relative path, then this directory is read relative to the *database-file* directory.

The -ad database option must be specified after the *database-file*, and applies only to that database.

*Example*

The database server applies the log files in the `backup` directory to the `mysample.db` database and then stops the database once the log files have been applied.

```
iqsrv15 "c:\mysample.db" -ad "c:\backup"
```

The database server applies the log files in the `backup` directory to the `mysample.db` database and the database continues running once the log files have been applied.

```
iqsrv15 "c:\mysample.db" -ad "c:\backup" -as
```

## -ar iqsrv15 database option

Specifies that any transaction log files located in the same directory as the current transaction log should be applied to the database.

*Syntax*

```
iqsrv15  [ server-options ] database-file -ar  ...
```

*Applies to*

All operating systems and database servers.

*Remarks*

When you include the -ar option, the database server looks for transaction log files associated with the database that are located in the same directory as the current transaction log. The transaction log location is obtained from the database. Transaction log files with starting log offsets greater than or equal to the start log offset stored in the database are applied, in log offset order. Once all the transaction log files have been applied, the database is stopped. You must also specify the -as option if you want the database to continue running once the transaction log files have been applied.

The -ar database option must be specified after the *database-file*, and applies only to that database.

*Example*

The database server applies the transaction log files (whose location is obtained from the database) to the `mysample.db` database. The database continues running after the transaction log files have been applied.

```
iqsrv15 "c:\mysample.db" -ar -as
```

## -as iqsrv15 database option

Specifies that the database should continue to run after transaction logs have been applied (used in conjunction with -ad or -ar).

*Syntax*

```
iqsrv15 [ server-options ] database-file { -ad log-dir | -ar } -
as  ...
```

*Applies to*

All operating systems and database servers.

*Remarks*

The -as option must be specified in conjunction with either the -ad or -ar option. When you include -as, the database continues running after the transaction logs are applied to it.

The -as database option must be specified after the *database-file*, and applies only to that database.

*Example*

The database server applies the transaction log files to the `mysample.db` database. In this case, because -ar is specified, the database server obtains the location of the transaction logs from the database. The database continues running after the log files have been applied.

```
iqsrv15 "c:\mysample.db" -ar -as
```

The database server applies the log files in the `backup` directory to the `mysample.db` database. The database continues running after the log files have been applied.

```
iqsrv15 "c:\mysample.db" -ad "c:\backup" -as
```

## -dh iqsrv15 Database Option

Makes a database undetectable when the Server Location utility **dblocate -d** is run against the server.

## -ds iqsrv15 database option

Specifies the directory where the dbspaces for the database and the transaction log are located.

*Syntax*

```
iqsrv15 database-file -ds dbspace-directory ...
```

*Applies to*

All operating systems and database servers.

*Remarks*

This option only applies to Catalog dbspaces, not IQ dbspaces.

When a dbspace directory is specified, the database server only searches this directory for dbspaces. The location of the dbspace appears in the database server messages window.

If your backup includes dbspaces with full path names, you can use this option to start the backed up copy of the database on the same computer as the original database while the original database is still running.

The -ds database option must be specified after the *database-file*, and applies only to that database.

If a transaction log file is not found in the directory specified by this option, then one is created in this location.

**Warning!** The -ds option should only be used for recovery. If you specify this option and the database has a current, live transaction log that is not located in the directory specified by the -ds option, then a new transaction log is created in the specified location.

*Example*

The following example starts a database server that looks for dbspaces in the directory `c:\backup\Nov15`:

```
iqsrv15 c:\backup\Nov15\my.db -ds c:\backup\Nov15\
```

The following example starts a database server that looks for dbspaces in the current directory:

```
iqsrv15 my.db -ds .
```

## -ek iqsrv15 Database Option

Specifies the database encryption key.

*Syntax*

**-ek** key

*Remarks*

Provided after the file name of a strongly encrypted database. Requires the key value as an argument to start an encrypted database. The key value is a string, including mixed cases, numbers, letters, and special characters. If you have a strongly encrypted database, you must provide the encryption key to use the database or transaction log; if you do not, the command fails. For a strongly encrypted database, you must specify either **-ek** or **-ep**, but not both.

## -iqfreq iqsrv15 Database Option

Marks the specified database as in use and restores the IQ portion of the database to its last known consistent state.

*Syntax*

**-iqfrec dbname**

*Remarks*

Do not use **-iqfrec** during normal operations; use it only while force-recovering a database after seeing s_buf or free list errors during recovery after IQ server failure. The *dbname* must be the physical database name, not a logical name or nickname.

**Note:** The option **-iqfrec** applies only to the IQ part of the database, not to the catalog store. **-iqfrec** does not enable a forced recovery on the SQL Anywhere part of the database (the catalog store).

Follow correct procedures when using **-iqfrec**. See *System Administration Guide: Volume 1 > System Recovery and Database Repair*.

## -m iqsrv15 Database Option

Truncates (deletes) the transaction log when a checkpoint is done, either at shutdown or as a result of a checkpoint scheduled by the server.

Truncation provides a way to automatically limit the growth of the transaction log. Checkpoint frequency is still controlled by the `CHECKPOINT_TIME` and `RECOVERY_TIME` options (also definable on the command line).

The **-m** option is useful where high volume transactions requiring fast response times are being processed, and the contents of the transaction log are not being relied upon for recovery or replication. When this option is selected, there is no protection provided against media failure on the device containing the database files.

To avoid database file fragmentation, Sybase recommends that you place the transaction log on a separate device or partition than the database itself.

This option is the same as the **-m** server option, but applies only to the current database or the database identified by the *database-file* command-line variable.

**Note:** Do not use the **-m** option with databases that are being replicated, as replication inherently relies on transaction log information. For this reason, never use the **-m** option on a multiplex database.

## -n iqsrv15 Database Option

Provides an alternate name, or nickname, for the database.

### *Syntax*

**-n** *name*

### *Remarks*

Using a nickname simplifies connections. For Open Client, the **-n** nickname must be the same as the entry in the `interfaces` file.

Since a database server can load several databases, the database name is used to distinguish the different databases. However, Sybase strongly recommends that you run only one database on an IQ server. If you must run two databases, start two IQ database servers on different ports.

By default, the database receives as a name the file name with the path and extension removed. For example, you start a server on `c:\sybase\IQ-15_3\demo\iqdemo.db` and do not specify the **-n** option, then the name of the database is **iqdemo**. To avoid using the default name, always specify a server name.

For naming conventions, see the **-n** server option.

**Note:** There are two -n switches. If **-n** does not follow a database file name, the option names the server. If **-n** appears after a database file name, the switch is a database switch.

## -r iqsrv15 Database Option

Forces all databases that start on the database server to be read-only.

No changes to the database are allowed: the database server doesn't modify the database file.

## -sm iqsrv15 Database Option

Provides an alternate database server name that can be used to access the read-only mirror database.

*Syntax*

**-sm** *name*

*Remarks*

The alternate-server-name is only active when the database server is acting as mirror for the database. By using the **-sm** and **-sn** command-line options, an application can always connect to the database on the primary or the mirror server, without knowing which physical server is acting as primary or mirror.

## -sn iqsrv15 Database Option

Provides an alternate server name for a single database running on a database server.

*Syntax*

**-sn** *name*

*Remarks*

The database server can be configured to listen for more than one server name for a particular database server. Server names other than the real server name are called alternate server names, and are specific to a particular database running on the database server. Clients using the alternate server name to connect can only connect to the database that specified the alternate server name. Alternate server names must be unique on the network; otherwise, the database fails to start. If the database is started in the server command and the alternate server name is not unique, the server fails to start.

## -xp iqsrv15 Database Option

Provides information to an operational server that allows it to connect to its partner and to the arbiter when database mirroring is being used.

*Syntax*

```
[server-options] database-file
```

```
-xp partner= (partner-conn);
```

```
auth=auth-str;
```

```
[ ;arbiter=(arbiter-conn)]
```

```
[ ;mode=|sync | async | page]
```

```
[ ;autofailover=[YES | NO]]
```

```
[ ;pagetimeout=n]
```

```
[ ;preferred=[YES | NO]]
```

### *Remarks*

The **-xp** database option must be specified after the database-file, and applies only to that database.

See *SQL Anywhere Server - Database Administration > Starting and connecting to your database > SQL Anywhere database server syntax > Database options > -xp dbsrv12 database option* .

**Note:** This reference points to SQL Anywhere documentation.

# stop_iq Database Shutdown Utility

On UNIX and Linux platforms, you can stop the database server using the **stop_iq** utility.

Use **stop_iq** to stop your server and close all user connections to your server.

**stop_iq** shuts down a server without regard for user connections or load process status. Normally, do not shut down a server while it is still connected to one or more clients.

**Note:** The **dbstop** utility offers a finer level of control, providing options to control whether a server is stopped based on user connections.

For full details on when to use **stop_iq**, see *System Administration Guide: Volume 1 > Sybase IQ Startup > Database Server Shutdown*.

## Stopping the Database Server with stop_iq

Run **stop_iq** from the command line.

1. Issue a command in this format:

   ```
   stop_iq [ -agent | -cleanup ] [ -stop [ one | all ] ] [ -user
   <user_name> ] [ -version [ 12 | 15 | all ]> ] [ -wait <seconds> ]
   ```
2. When prompted to stop your server, respond Y (yes).
3. If the server is connected to one or more clients, respond to the warning that uncommitted transactions will be lost.

The following example illustrates how to use **stop_iq** interactively to list all the servers that are running, and then shut down one server.

```
% stop_iq
```

```
Checking system ...
The following 1 server(s) are owned by 'TEST'

##     Owner    PID   Started  CPU_Time   Additional Information
 -- ---------  -----  --------  --------  -----------------------
1:   TEST     22399  08:56:39     1:43  SVR:QA_sun7qa DB:iqdemo PORT:
8888
/sun7qa1/users/QA/090513/IQ-15_4/bin64/iqsrv15 @iqdemo.cfg iqdemo.db
-ti 4400
--
    Please note that 'stop_iq' will shutdown a server completely
    without regard for users connections or load processes status.
    For a finer level of detail the utility 'dbstop' has the options
    to control whether a server is stopped based on active
connections.

Do you want to stop the server displayed above <Y/N>? Y
```

```
Shutting down server (22399) ...
Checkpointing server (22399) ...
Server shutdown.
```

## stop_iq Options

This table lists the options available for the **stop_iq** utility.

**Table 33. stop_iq options**

| Switch | Description |
|---|---|
| **-agent** | Stops the IQ Agent on UNIX or Linux systems. |
| **-cleanup** | Removes the orphan IQ process on Linux. |
| **-stop [ one \| all ]** | Removes user interaction with **stop_iq**. Assumes a yes response to all questions. |
| **-user** | Performs two functions:<br>• If, due to truncation or substitution, **stop_iq** cannot find the server or agent owned by the current user, the system manager can specify the name and id found in the process table in the **-user** argument to shut down the server/ agent with the **stop_iq** utility.<br>• A user with root privileges can shut down another user's server or agent without having to log in as that user. The **stop_iq** utility has no superuser (su) or root powers, so a non-privileged user cannot shut down a server owned by another user. |
| **-version** | Specifies the major version of Sybase IQ that is being used. The default is the current major version (*15*). |
| **-wait** | Specify the time to wait for the server to shut down before timeout expires. |

## Stopping Servers in a cron or at Job

If using **stop_iq** in a **cron** or **at** job, use the appropriate **-stop** option:

**Table 34. stop_iq -stop options**

| Option | Description |
|---|---|
| -stop one | Shuts down a single server, when exactly one running server was started by the user ID that starts the **cron** or **at** job. This prevents the wrong server from being shut down if several are running. |
| -stop all | Shuts down all servers that were started by the user ID that starts the server. |

For example:

```
stop_iq -stop one
```

```
stop_iq -stop all
```

**Note:** In a **cron** statement, you must specify the complete path name to the **stop_iq** executable.

## Servers with Long Paths

Depending on the operating system, the **stop_iq** utility may fail to report running servers when the path to the server exceeds 74 characters.

As a workaround, use a **ps -ef** command to display servers with long paths. For example:

```
ps -ef|grep myserver

rsmithson 1133     1   0 07:04:32 ?         223:35
/sunsys1234/users/rsmithson/mybigtest1234_withdeletion
_allcol/IQ-15_4/bin64/myserver

rsmithson 2046   862   0 10:02:30 pts/3       0:00 grep
myserver
```

# Appendix: dbisqlc Interactive SQL Classic Utility (Deprecated)

The Interactive SQL Classic (**dbisqlc**) utility executes SQL statements against a database. This utility is deprecated.

The utility is similar to the Interactive SQL utility, except that it is not implemented in Java. The absence of Java can be an advantage if you are deploying the utility to a computer with limited resources.

**Note:** Interactive SQL Classic is deprecated; however, there are currently no plans to remove it. Interactive SQL Classic is provided for backwards compatibility for running SQL scripts and as a lightweight tool for deployment. It does not support all the features that Interactive SQL supports and may not support all the features available in the current version of Sybase IQ. It is recommended that you use the Interactive SQL utility.

The Interactive SQL Classic utility is supported on Microsoft Windows and UNIX.

**See also**

- *dbisql Interactive SQL Utility* on page 13
- *iqisql Interactive SQL Utility* on page 105
- *isql Interactive SQL Utility* on page 113

## Interactive SQL Classic Syntax

Invoke Interactive SQL Classic from a command prompt.

```
dbisqlc [ options ] [ dbisqlc-command  | command-file ]
```

If you specify *dbisqlc-command*, Interactive SQL Classic executes the command. You can also specify a command file name. If you do not specify *dbisqlc-command* or *command-file* argument, Interactive SQL Classic enters interactive mode, where you can type a command into a command window.

## Interactive SQL Classic Options

Specify these options when invoking Interactive SQL Classic from a command prompt.

**Table 35. Interactive SQL Classic options**

| Option | Description |
|---|---|
| **-c** *"keyword=value; ..."* | Specifies connection parameters. See *System Administration Guide: Volume 1 > Connection and Communication Parameters*. If you do not specify any connection parameters, the environment variable SQLCONNECT is used. If Interactive SQL cannot connect, enter the appropriate parameters in the dialog box that appears.<br><br>**Note:** Sybase recommends that you always specify connection parameters for **Interactive SQL Classic** instead of relying on defaults, whether you specify them in a command line or an initialization file such as `.odbc.ini` on UNIX, or `odbc.ini` on Windows. If you start more than one database on a server, for example, specify the database name, and in a network with subnets, specify the communications protocol parameter with host number. See *System Administration Guide: Volume 1 > Sybase IQ Connections*. |
| **-d** *delimiter* | Specifies a command delimiter. By default, the delimiter is the semicolon. |
| **-q** | Quiet mode—does not display output messages. This option is useful only if you start Interactive SQL Classic with a command or command file. |
| **-r** | Returns the error `“Not enough fields allocated in sqlda”` if the defined result set of the stored procedure does not match the actual result set. This option may be useful when you are querying stored procedures. |
| **-x** | Checks syntax only. Scans commands but does not execute them. You may find this option useful for checking long command files for syntax errors. |

## Function and Special Keys (UNIX)

Use function keys and special keys to move data and list database tables.

**Table 36. Interactive SQL Classic Function and Special Keys on UNIX**

| Function key | Description |
|---|---|
| F5 | Move data to the left by one column in the data window. |

| Function key | Description |
|---|---|
| Shift+F5 | Move data to the left by one character. |
| F6 | Move data to the right by one column. |
| Shift+F6 | Move data to the right by one character. |
| F7 | Display a list of the tables in the database. Use the up and down arrow keys to scroll through the table names changing the highlighted table name. While the list appears, press Enter to insert the current table name into the command window at the cursor position. While the list appears, press F7 to see a list of columns for the highlighted table. Again, use Enter to select the highlighted column name and put it into the command window at the cursor position. |
| Ctrl+PgUp | Move to the top of data. |
| Ctrl+PgDn | Move to bottom of data. |

## Function and Special Keys (Windows)

Use function keys and special keys to move data and list database tables.

**Table 37. Interactive SQL Classic Function and Special Keys on Windows**

| Function key | Description |
|---|---|
| F5 | Move data to the left by one column in the data window. |
| Shift+F5 | Move data to the left by one character. |
| F6 | Move data to the right by one column. |
| Shift+F6 | Move data to the right by one character. |
| F7 | Display a list of the tables in the database. Use the up and down arrow keys to scroll through the table names changing the highlighted table name. While the list appears, press Enter to insert the current table name into the command window at the cursor position. While the list appears, press F7 to see a list of columns for the highlighted table. Again, use Enter to select the highlighted column name and put it into the command window at the cursor position. |
| F9 | Execute the command that is in the command window. This operation can also be performed with the mouse by clicking Execute. |
| F10 | Activate the menus at the top of the window. |
| Page Up | Move data up a page. |
| Page Down | Move data down a page. |

| Function key | Description |
|---|---|
| Ctrl+PageUp | Move to top of data. |
| Ctrl+PageDown | Move to bottom of data. |

## Command Recall Keys (Windows)

Use key sequences to recall previous commands

**Table 38. Interactive SQL Classic recall keys**

| Key sequence | Description |
|---|---|
| Ctrl+r | Brings up the command recall window |
| Ctrl+p | Cycles backwards through previously executed commands. Retrieved commands are placed into the command window |
| Ctrl+n | Cycles forward through previously executed commands |

# Index