Command Interface Guide

Version 2.0.0

Table of Contents

License Statement

Acknowledgements

Version	Date
1.3.0	January, 2016

Chapter 1. About This Document

This guide describes how to use the Trafodion Command Interface (TrafCI) on a client workstation to connect to and query a Trafodion database. The TrafCI enables you to run SQL statements interactively or from script files.

1.1. Intended Audience

This guide is intended for database administrators and support personnel who are maintaining and monitoring a Trafodion database.

1.2. New and Changed Information

This manual shows updated versions for Trafodion Release 1.3.0.

1.3. Notation Conventions

This list summarizes the notation conventions for syntax presentation in this manual.

- UPPERCASE LETTERS

  Uppercase letters indicate keywords and reserved words. Type these items exactly as shown. Items not enclosed in brackets are required.

  ```
  SELECT
  ```

- lowercase letters

  Lowercase letters, regardless of font, indicate variable items that you supply. Items not enclosed in brackets are required.

  ```
  file-name
  ```

- [ ] Brackets

  Brackets enclose optional syntax items.

```
DATETIME [start-field TO] end-field
```

  A group of items enclosed in brackets is a list from which you can choose one item or none.

  The items in the list can be arranged either vertically, with aligned brackets on each side of the list, or horizontally, enclosed in a pair of brackets and separated by vertical lines.

  For example:

```
DROP SCHEMA schema [CASCADE]
DROP SCHEMA schema [ CASCADE | RESTRICT ]
```

- { } Braces

  Braces enclose required syntax items.

```
FROM { grantee [, grantee ] ... }
```

  A group of items enclosed in braces is a list from which you are required to choose one item.

  The items in the list can be arranged either vertically, with aligned braces on each side of the list, or horizontally, enclosed in a pair of braces and separated by vertical lines.

  For example:

```
INTERVAL { start-field TO end-field }
{ single-field }
INTERVAL { start-field TO end-field | single-field }
```

- | Vertical Line

  A vertical line separates alternatives in a horizontal list that is enclosed in brackets or braces.

```
{expression | NULL}
```

- … Ellipsis

  An ellipsis immediately following a pair of brackets or braces indicates that you can repeat the enclosed sequence of syntax items any number of times.

```
ATTRIBUTE[S] attribute [, attribute] ...
{, sql-expression } ...
```

  An ellipsis immediately following a single syntax item indicates that you can repeat that syntax item any number of times.

  For example:

```
expression-n ...
```

- Punctuation

  Parentheses, commas, semicolons, and other symbols not previously described must be typed as shown.

```
DAY (datetime-expression)
@script-file
```

  Quotation marks around a symbol such as a bracket or brace indicate the symbol is a required character that you must type as shown.

  For example:

```
"{" module-name [, module-name] ... "}"
```

- Item Spacing

  Spaces shown between items are required unless one of the items is a punctuation symbol such as a parenthesis or a comma.

  ```
  DAY (datetime-expression) DAY(datetime-expression)
  ```

  If there is no space between two items, spaces are not permitted. In this example, no spaces are permitted between the period and any other items:

  ```
  myfile.sh
  ```

- Line Spacing

  If the syntax of a command is too long to fit on a single line, each continuation line is indented three spaces and is separated from the preceding line by a blank line.

  This spacing distinguishes items in a continuation line from items in a vertical list of selections.

  ```
  match-value [NOT] LIKE _pattern
     [ESCAPE esc-char-expression]
  ```

1.4. Publishing History

Product Version	Publication Date
Trafodion Release 1.3.0	To be announced.

1.5. Comments Encouraged

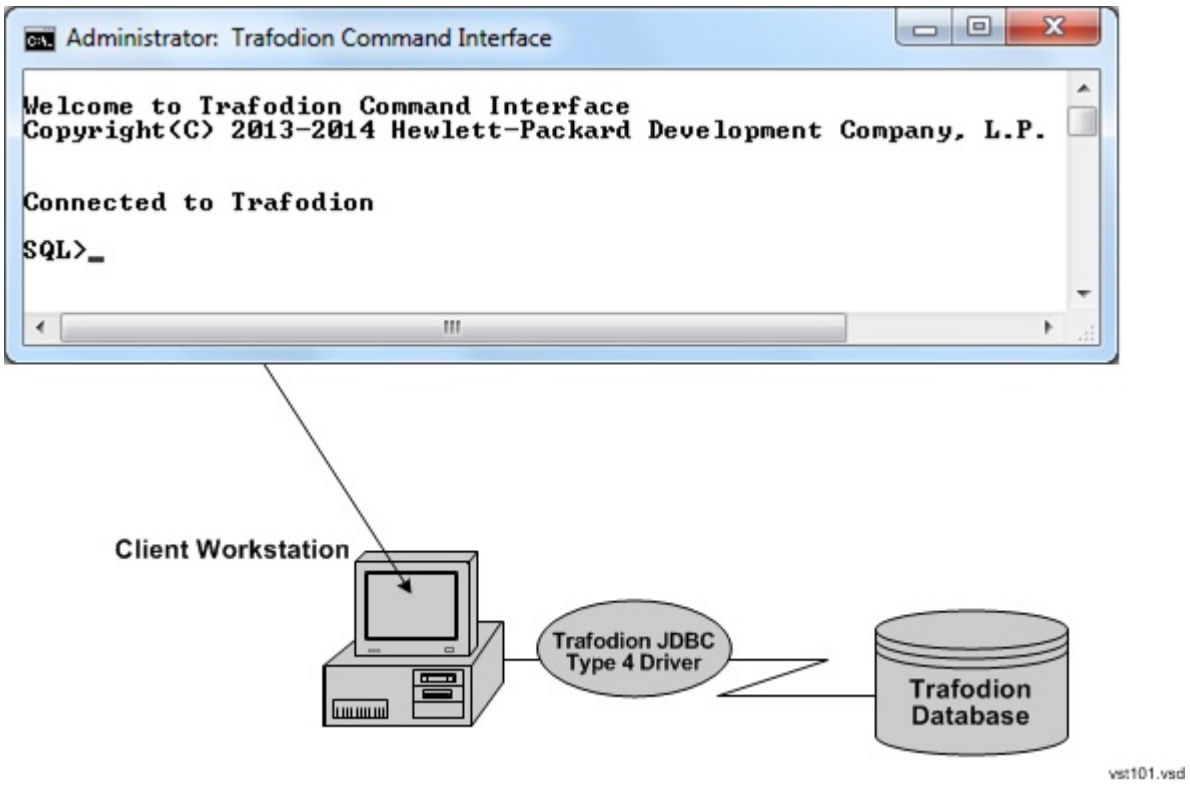
The Trafodion community encourages your comments concerning this document. We are committed to providing documentation that meets your needs. Send any errors found, suggestions for improvement, or compliments to:

issues@trafodion.incubator.apache.org

Include the document title and any comment, error found, or suggestion for improvement you have concerning this document. Or, even better, join our community and help us improve our documentation. Please refer to Trafodion Contributor Guide for details.

Chapter 2. Introduction

The Trafodion Command Interface (TrafCI) is a command-line interface that you download and install on a client workstation that has the Trafodion JDBC Type 4 Driver installed. Operating systems that support the JDBC driver include Windows and Linux. The JDBC driver connects TrafCI on a client workstation to a Trafodion database.



TrafCI enables you to perform daily administrative and database management tasks by running SQL statements or other commands interactively or from script files. You can also run TrafCI from a Perl or Python command line or from Perl or Python programs.

Chapter 3. Install and Configure

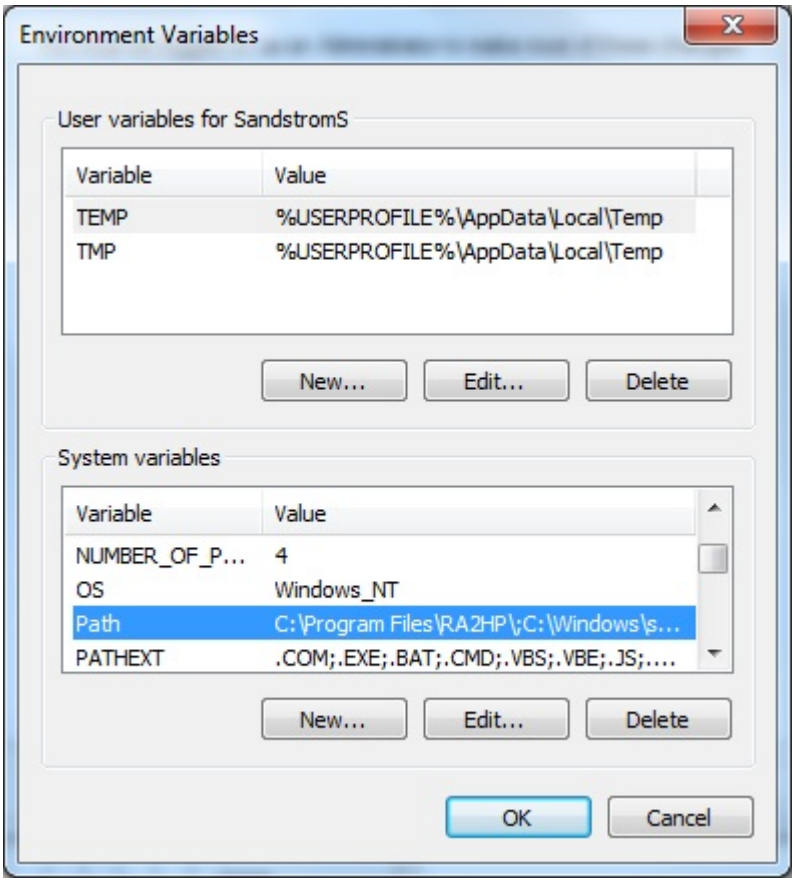
3.1. Install TrafCI

To install TrafCI on a client workstation, follow the procedures in the *Trafodion Client Installation Guide*.

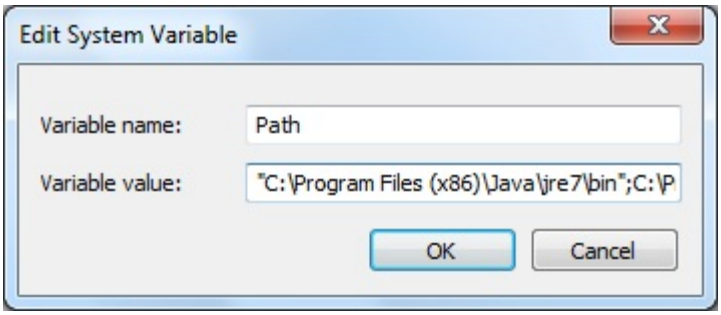
3.2. Verify and Set the Java Path

3.2.1. Set PATH on Windows

1. Right-click the **Computer** icon on your desktop, and then select **Properties**. The **Control Panel>System and Security>System** window appears.
2. In the left navigation bar, click the **Advanced** system settings link.
3. In the **System Properties** dialog box, click the **Environment Variables** button.
4. Under *System variables*, select the variable named **Path**, and then click **Edit**:



5. Place the cursor at the beginning of the *Variable* value field and enter the path of the Java `bin` directory, ending with a semicolon (`;`):




```
"C:\Program Files (x86)\Java\jre7\bin";
```

Check that no space exists after the semicolon (;) in the path. If there are spaces in the directory name, delimit the entire directory path in double quotes (") before the semicolon.

6. Click **OK**.
7. Verify that the updated Path appears under *System variables*, and click OK.
8. In the **System Properties** dialog box, click **OK** to accept the changes.

3.2.2. Set PATH on Linux

1. Open the user profile (`.profile` or `.bash_profile` for the Bash shell) in the `$HOME` directory.

```
vi .profile
```

2. In the user profile, set the `PATH` environment variable to include path of the Java bin directory.

```
export PATH=/opt/java1.7/jre/bin:$PATH
```

Place the path of the Java bin directory before `$PATH`, and check that no space exists after the colon (`:`) in the path. In the C shell, use the `setenv` command instead of export.

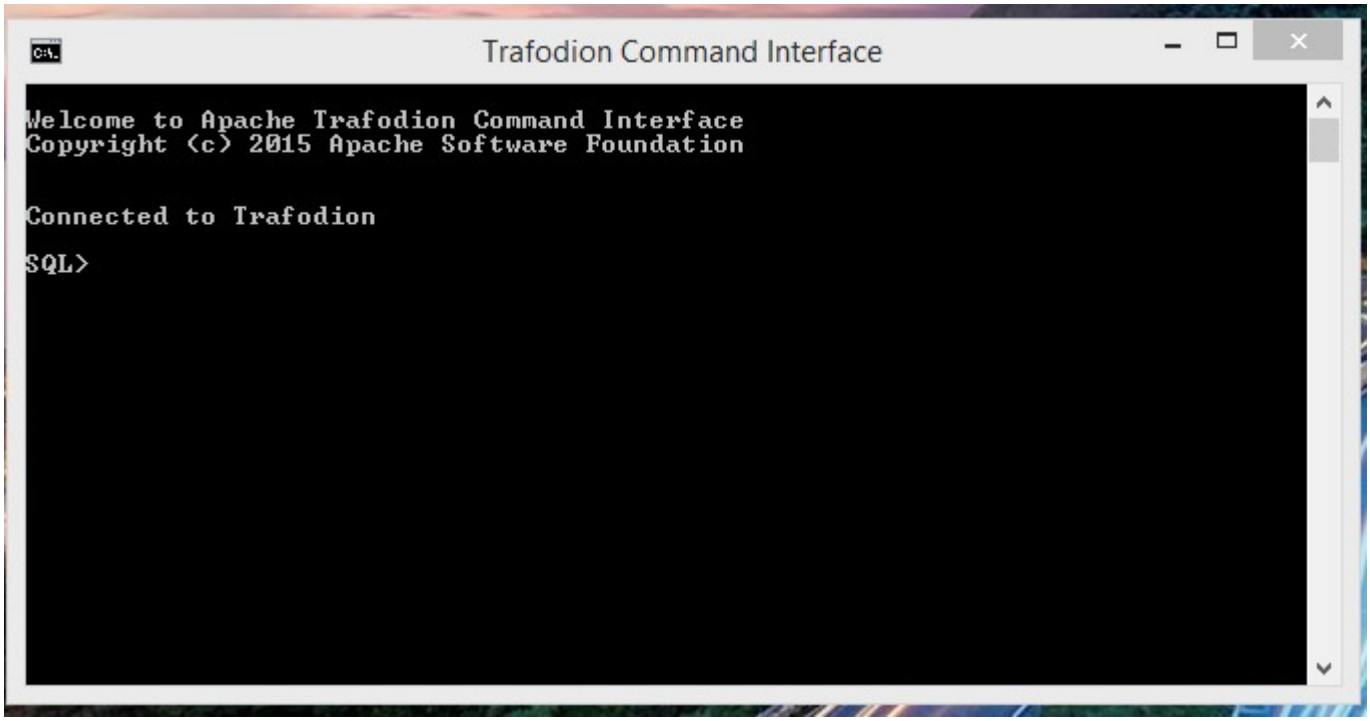
3. To activate the changes, either log out and log in again or execute the user profile.

```
. .profile
```

3.3. Test TrafCI Launch

1. Launch TrafCI and verify that you can connect to the database. For instructions, see Launch TrafCI.

   This window should appear:



2. If you cannot launch TrafCI or connect to the database, verify that:
   - The database platform is available and running, and the port number is correct for the database platform.
   - The Java path is set to the correct location. See Verify and Set the Java Path.
   - You installed the TrafCI and JDBC software files correctly.

See the *Trafodion Client Installation Guide*.

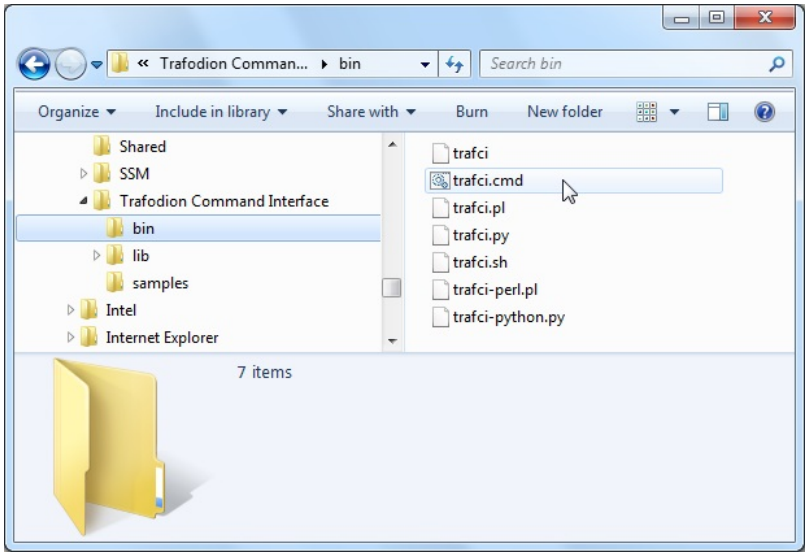
Chapter 4. Launch TrafCI

This chapter describes how to launch TrafCI from the Window or Linux environment of a client workstation. For information about launching TrafCI from Perl or Python, see Run TrafCI from Perl or Python.

Before launching TrafCI, make sure that you have set the Java path to the correct location. See Verify and Set Java Path.

4.1. Launch TrafCI on Windows Workstation

1. Find the Windows launch file, `trafci.cmd`, in the `bin` folder:



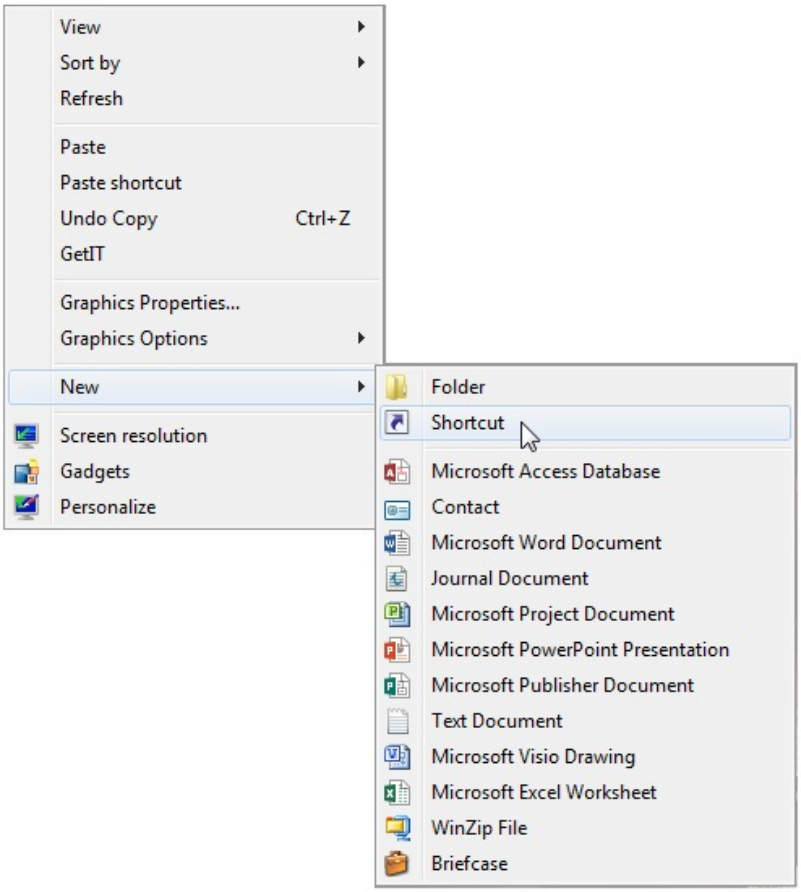
2. Double-click the `trafci.cmd` file.

   TrafCI appears, prompting you to enter the host name or IP address of the database platform, your user name, and password. See Log In to Database Platform.

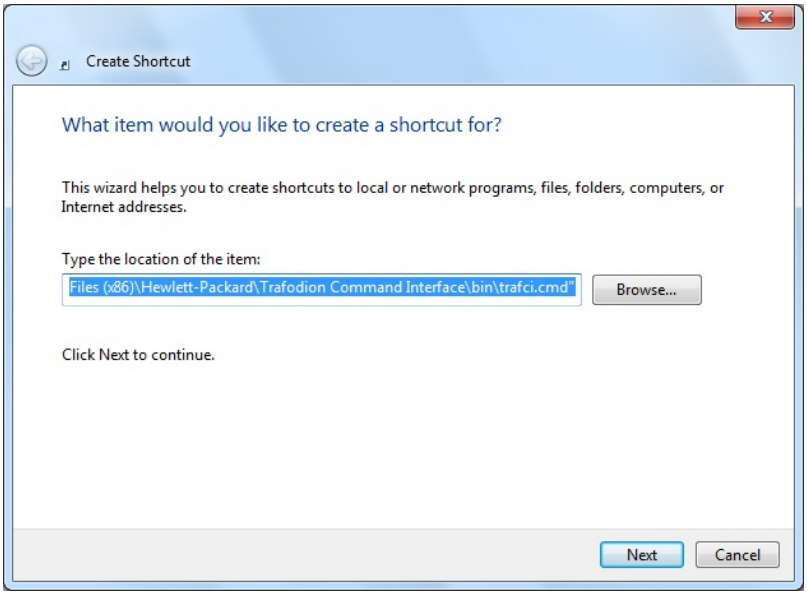
4.1.1. Create `trafci.cmd` Shortcut

To enable a user to launch TrafCI from a shortcut icon on the desktop:

1. Right-click the desktop and select **New>Shortcut**:

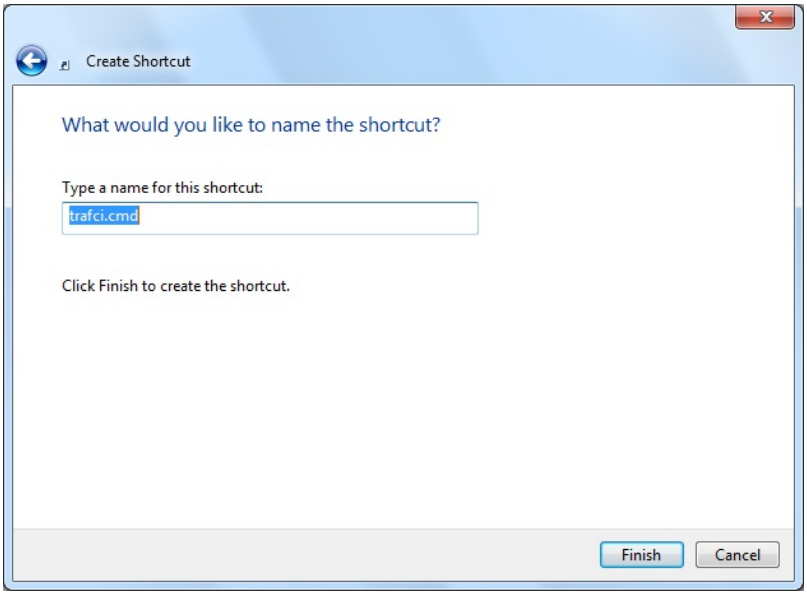


2. Type the location of `trafci.cmd` within double quotes (") or click **Browse** to locate that file, and then click **Next**:



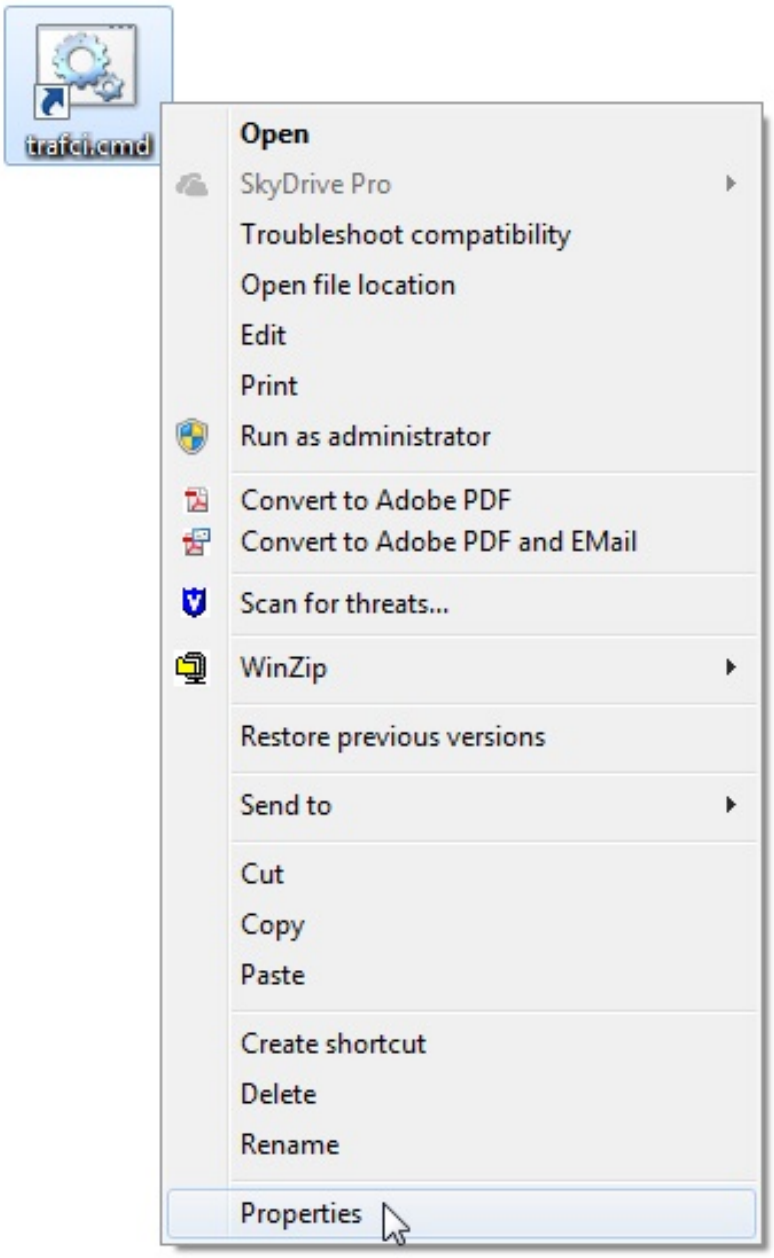
For the locations of the installed TrafCI software files, see the *Trafodion Client Installation Guide*.

3. Type a name for the shortcut and click **Finish**:



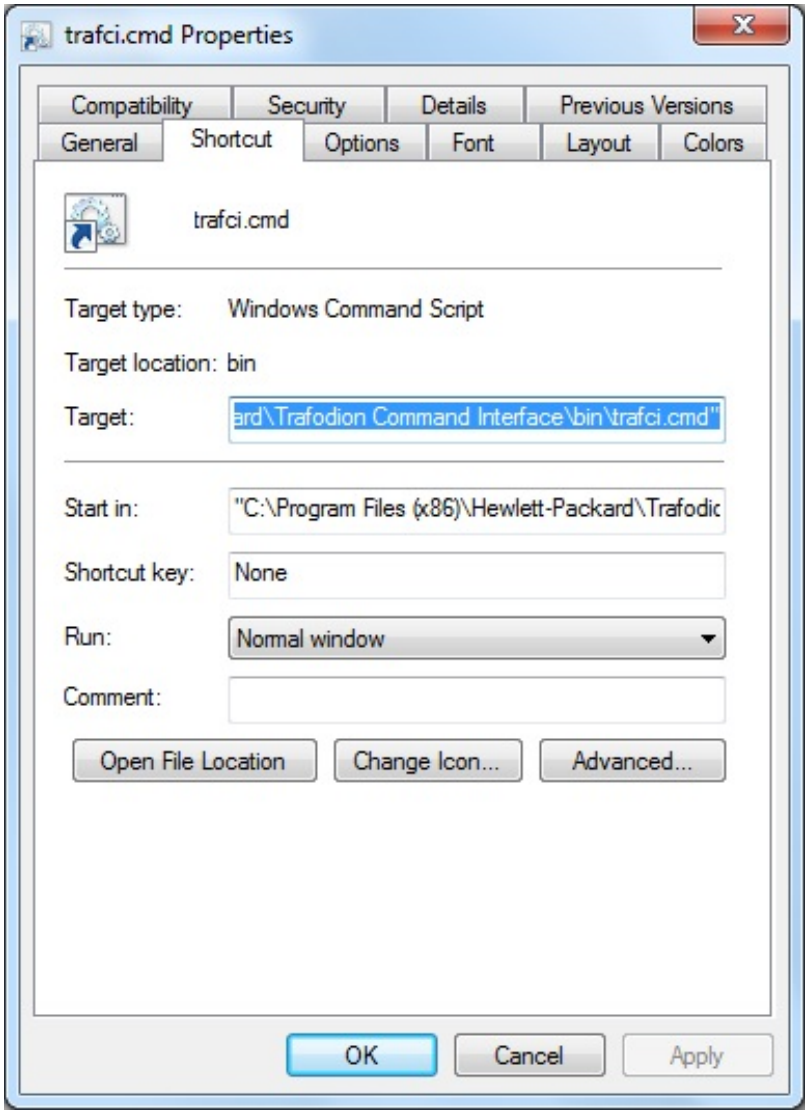
4. If desired, specify optional launch parameters for the shortcut:

   a. Right-click the shortcut icon and select **Properties**:



   b. Select the **Shortcut** tab.

   c. In the **Target** box, insert a space after "…`\Trafodion Command Interface\bin\trafci.cmd`" and add the optional launch parameters:



For more information, see Optional Launch Parameters.

d. Click **OK**.

5. To launch TrafCI, double-click the shortcut icon.

TrafCI appears. If you did not set the optional launch parameters, TrafCI prompts you to enter the host name or IP address of the database platform, your user name, and password. See Log In to Database Platform.

4.2. Launch TrafCI on Linux Workstation

In the terminal window, enter:

```
./<trafci-installation-directory>/trafci/bin/trafci.sh
```

<trafci-installation-directory> is the directory where you installed the TrafCI software files. For more information, see the *Trafodion Client Installation Guide*.

4.2.1. Set `trafci.sh` PATH

To enable a user to launch TrafCI anywhere on the client workstation:

1. Open the user profile (`.profile` or `.bash_profile` for the Bash shell) in the `$HOME` directory.

   ```
   cd $HOME
   vi .profile
   ```

2. In the user profile, set the PATH environment variable to include the path of the `trafci.sh` file.

   ```
   export PATH=/<trafci-installation-directory>/trafci/bin/: ...
   ```

   trafci-installation-directory is the directory where you installed the TrafCI software files. For more information, see the *Trafodion Client Installation Guide*. Check that no space is after the colon (`:`) in the path.

   In the C shell, use the `setenv` command instead of `export`.

3. To activate the changes, either log out and log in again or execute the user profile.

   ```
   . .profile
   ```

4. On the command line, execute the `trafci.sh` file to launch TrafCI:

```
trafci.sh
```

TrafCI appears, prompting you to enter the host name or IP address of the database platform, your user name, and password. See Log In to Database Platform.

To enable all users to launch TrafCI anywhere on the system, create a symbolic link to the `trafci.sh` file in the `/usr/bin` or `/usr/local/bin` directory:

```
ln -s ./<trafci-installation-directory>/trafci/bin/trafci.sh /usr/bin/trafci.sh
```

4.2.2. Preset the Optional Launch Parameters

To preset the optional launch parameters for each session, use an `alias` in the shell command.

```
alias trafci='trafci.sh -h 16.123.456.78:37800 -u user1 -p xxxxxx'
```

You can add the alias, trafci, to the user profile, or you can enter it at a command prompt. For more information about the optional launch parameters, see Use Optional Launch Parameters.

4.3. Log In to Database Platform

4.3.1. Log In Without Login Parameters

If you launch TrafCI and do not specify login parameters on the command line, follow these steps:

1. After you launch TrafCI, TrafCI shows the welcome banner and prompts you to enter the host name or IP address of the database platform:

   ```
   Host Name/IP Address: _
   ```

   Enter a host name:

   ```
   host-name[.domain-name][:port-number]
   ```

   - If you do not specify the domain name, TrafCI uses the domain of the client workstation.
   - If you do not specify a port number, TrafCI uses the default port umber, which is `37800`.

     Or enter an IP address:

     ```
     IP-address[:port-number]
     ```

2. Enter your directory-service (or LDAP) user name. User names are case-insensitive.
3. Enter your password. Passwords are case-sensitive.
4. After you finish logging in to the database platform, the SQL prompt appears:

   ```
   Connected to Trafodion

   SQL>
   ```

At the prompt, you can enter an SQL statement or an interface command. For more information, see Run Interactive Commands in TrafCI.

TrafCI allows you to reenter the login values, with a maximum of three retries, before it closes the session. For more information, see Retry Login.

4.3.2. Use Login Parameters

To avoid entering a host name, user name, or password each time you launch TrafCI, use these login parameters:

- `-h` or `-host`
- `-u` or `-user`
- `-p` or `-password`

Example: Windows Login

```
cd <trafci-installation-directory>\Trafodion Command Interface\bin

trafci.cmd -h 16.123.456.78:37800 -u user1 -p xxxxxx
```

Example: Linux Login

```
cd <trafci-installation-directory>/trafci/bin

./trafci.sh -h 16.123.456.78:37800 -u user1 -p xxxxxx
```

TrafCI launches and prompts you to enter an SQL statement or an interface command:

```
Welcome to Trafodion Command Interface
Copyright(C) 2013–2105 Apache Software Foundation

Connected to Trafodion

SQL>
```

For more information about the login parameters, see Use Optional Launch Parameters.

You can include these parameters in a shortcut to the `trafci.cmd` file or in a launch file for the `trafci.sh` file. For more information, see Create `trafci.cmd` Shortcut or Preset the Optional Launch Parameters, respectively.

4.4. Retry Login

TrafCI allows you to reenter the login values, with a maximum of three retries, before it closes the session.

TrafCI applies the retry logic as follows:

- If you specify an invalid host name, TrafCI prompts you to reenter the host name.

  Example

```
$ trafci –h dd # dd is invalid

Welcome to Trafodion Command Interface
Copyright(C) 2013–2105 Apache Software

Unknown Host: dd

Host Name/IP Address: 172.16.1.1

User Name: user1
Password:

Connected to Trafodion SQL>
```

- If you specify an invalid user name or password, TrafCI prompts you to reenter the user name and password.

  If you specify an invalid password, TrafCI prompts only for your user name and password. After three unsuccessful retries, the session is terminated:

  Example

```
$ trafci –h 172.16.1.1 –u user1 –p x

Welcome to Trafodion Command Interface
Copyright(C) 2013–2105 Apache Software

**** ERROR[8837] CLI Authentication : User: user1 : invalid username or password
[2105-03-12 16:23:44]

User Name: user1
Password:

**** ERROR[8837] CLI Authentication : User: user1 : invalid username or password
[2105-03-12 16:25:28]

User Name: user1
Password:

**** ERROR[8837] CLI Authentication : User: user1 : invalid username or password
[2105-03-12 16:26:36]

Press any key to close this session
```

- If all the login parameters that you specify are invalid, TrafCI prompts you to enter the host name. When you enter a valid host name or IP address, TrafCI prompts you to enter a user name and password.
- The retry logic applies to the CONNECT and RECONNECT commands. For the RECONNECT command, the retry logic applies only when no prior connection has been established (`-noconnect`).

  For example, if you specify the CONNECT command with a valid user name and host name, then TrafCI prompts for the user name and password only.

```
SQL> connect user1/xxx@172.16.1.1

com.hp.jdbc.HPT4Exception: **** ERROR[8837] CLI Authentication : User: user1 :
invalid username or password [2105-03-12 16:35:15]

User Name: user1
Password: abc

Connected to Trafodion SQL>
```

- TrafCI does not prompt you to reenter the login values in these cases:
- When you include the `-q` or `-version` parameter on the command line. (The `-s` parameter permits login retries.)
  - For a session started using redirected or piped input.

In these cases, TrafCI returns an error message and closes the session. You must re-launch the TrafCI session to connect to the Trafodion database.

4.5. Optional Launch Parameters

To customize how you launch and log in to TrafCI, use the optional parameters described in the table below on the command line:

```
trafci{.sh | .cmd} [optional-parameter]...
```

- *optional-parameter*

  is one of the launch or login parameters. For details, see the following table.

Launch or Login Parameter	Description
{-h | -host} host-name[:port-number] {-h | -host} IP-address[:port-number]	Specifies the host name or IP address of the database platform to which you want the client to connect. The *host-name* should include the domain name of the database platform if it is different from the domain of the client workstation. If you do not specify a port number, TrafCI uses the default port number, which is 37800. See Use Login Parameters.
{-u | -user} *username*	Specifies the user name for logging in to the database platform. The *username* is case-insensitive. For an example, see Use Login Parameters.
{-r | -role} *role-name*	Reserved for future use.
{-p | -password} *password*	Specifies the password of the user for logging in to the database platform. *password* is case-sensitive. For an example, see Use Login Parameters.
{-q | -sql} "*command*"	Specifies that an SQL statement or an interface command be run when launching TrafCI. You cannot specify this parameter at the same time as the -s or -script parameter. For more information, see Run Command When Launching TrafCI.
{-s | -script} *script-file-name*	Specifies that a script file be run when launching TrafCI in interactive mode. You cannot specify this parameter at the same time as the -q or -sql parameter. For more information, see Run Script When Launching TrafCI.
-noconnect	Launches an TrafCI session without connecting to the database. For more information, see Launch TrafCI Without Connecting to the Database.

Launch or Login Parameter	Description
`-version`	Displays the build version of TrafCI and the Trafodion JDBC Type 4 Driver. Upon completion of the display, the client exits. If any other parameters are included with the `-version` parameter, they are ignored. For more information, see Run TrafCI With `-version`.
`-help`	Displays a list of accepted arguments with descriptions and then exits. For more information, see Run TrafCI With `-version`.

4.6. Run Command When Launching TrafCI

To execute an SQL statement or an interface command when launching TrafCI, use the `-q` or `-sql` command-line parameter. This parameter enables you to run a single command on the command line without having to enter commands in TrafCI.

You cannot specify this parameter at the same time as the `-s` or `-script` parameter.

When using `-q` or `-sql`, you must enclose the command in double quotes ("). The SQL terminator is not required at the end of an SQL statement and is disallowed after an interface command.

Although you can run any of the interface commands with `-q` or `-sql`, the `@`, `OBEY`, and `PRUN` commands are the most useful.

Example

Use `-q` or `-sql` with the `CREATE SCHEMA` statement to create a schema when launching TrafCI:

- On Windows, in the **Command Prompt** window, enter:

```
cd _trafci-installation-directory_\Trafodion Command Interface\bin
trafci.cmd -q "create schema persnl"
```

- On Linux or UNIX, in the terminal window, enter:

```
cd _trafci-installation-directory_/trafci/bin
./trafci.sh -q "create schema persnl"
```

After you enter the SQL statement, TrafCI launches and prompts you to log in by default (if you did not specify `-h`, `-u`, and `-p` on the command line), runs the SQL statement, and then returns to the command prompt:

```
Host Name/IP Address: 16.123.456.78:37800 User Name: user1

Password:

--- SQL operation complete.

C:\Program Files (x86)\Apache Software Foundation\Trafodion Command Interface\bin>
```

Example

Use `-q` or `-sql` with the PRUN command to run multiple script files simultaneously from the command line:

- On Windows, in the **Command Prompt** window, enter:

```
cd <trafci-installation-directory>\Trafodion Command Interface\bin
trafci.cmd -q "prun"
```

- On Linux, in the terminal window, enter:

```
cd <trafci-installation-directory>/trafci/bin
./trafci.sh -q "prun"
```

After you enter the interface command, TrafCI launches and prompts you to log in by default (if you did not specify `-h`, `-u`, and `-p` on the command line), and runs the command.\ The parallel run (PRUN) operation prompts you to enter settings and then executes the script files. At the end of the PRUN operation, TrafCI returns to the command prompt.

For more information about the PRUN operation, see PRUN Command.

4.7. Run Script When Launching TrafCI

To run a script file when launching TrafCI, use the `-s` or `-script` command-line parameter.

You cannot specify this parameter at the same time as the `-q` or `-sql` parameter.

After you launch TrafCI with `-s` or `-script`, TrafCI executes the script file in interactive mode. TrafCI remains open until you enter the `EXIT`, `QUIT`, or `DISCONNECT` command. To quit the interface immediately after executing a script file, include the `EXIT`, `QUIT`, or `DISCONNECT` command at the end of the script file.

Example

You can create a script file that contains `SET` commands that customize a session when you launch TrafCI:

For more information, Create a Script File.

Example

- On Windows, in the **Command Prompt** window, enter:

```
cd <trafci-installation-directory>\Trafodion Command Interface\bin
trafci.cmd -s settings.txt
```

  Specify the full path of the script file if it is outside the directory of `trafci.cmd`.

- On Linux, in the terminal window, enter:

```
cd <trafci-installation-directory>/trafci/bin +
./trafci.sh -s settings.txt
```

  Specify the full path of the script file if it is outside the directory of `trafci.sh`.

TrafCI launches and prompts you to log in by default (if you did not specify `-h`, `-u`, and `-p` on the command line), and runs the commands in the script file:

```
Welcome to Trafodion Command Interface
Copyright(C) 2013–2105 Apache Software Foundation

Host Name/IP Address: 16.123.456.78:37800 User Name: user1
Password:
Connected to Trafodion

SQL>SET IDLETIMEOUT 0

SQL>SET SQLPROMPT *

*SET TIME ON

14:14:57 *SET TIMING ON

2:14:57 PM *SET SQLTERMINATOR .
```

4.8. Launch TrafCI Without Connecting to the Database

To start TrafCI without connecting to a Trafodion database, use the `-noconnect option`. See `DISCONNECT` command for a list of interface commands that can be run without a connection.

Example

- On Windows, in the **Command Prompt** window, enter:

```
cd <trafci-installation-directory>\Trafodion Command Interface\bin
trafci.cmd -noconnect
```

- On Linux, in the terminal window, enter:

```
cd <trafci-installation-directory>/trafci/bin
./trafci.sh -noconnect
```

4.9. Run TrafCI With `-version`

To display the build version of TrafCI and the Trafodion JDBC Type 4 Driver, use the `-version` option. If other parameters are included with the `-version` parameter, they are ignored.

Example

- On Windows, in the **Command Prompt** window, enter:

```
cd <trafci-installation-directory>\Trafodion Command Interface\bin
trafci.cmd -version
```

- On Linux, in the terminal window, enter:

```
cd <trafci-installation-directory>/trafci/bin
./trafci.sh -version
```

```
Welcome to Trafodion Command Interface
Copyright(C) 2013–2105 Apache Software Foundation

Trafodion JDBC Type 4 Driver : Traf_JDBC_Type4_Build_40646 Trafodion
Command Interface : TrafCI_Build_40646
```

4.10. Run TrafCI With -help

To display a list of acceptable list of parameters, including proper usage information, use the `-help` option. After displaying this information the application exits.

Example

- On Windows, in the **Command Prompt** window, enter:

```
cd <trafci-installation-directory>\Trafodion Command Interface\bin
trafci -help
```

- On Linux, in the terminal window, enter:

```
cd <trafci-installation-directory>/trafci/bin
./trafci.sh -help
```

4.11. Exit TrafCI

To exit TrafCI, enter one of these commands at a prompt:

- `EXIT`
- `QUIT`

Example

```
SQL> QUIT
```

These commands are not case-sensitive and do not require a terminator before you press **Enter**. After you enter one of these commands, TrafCI immediately quits running and disappears from the screen.

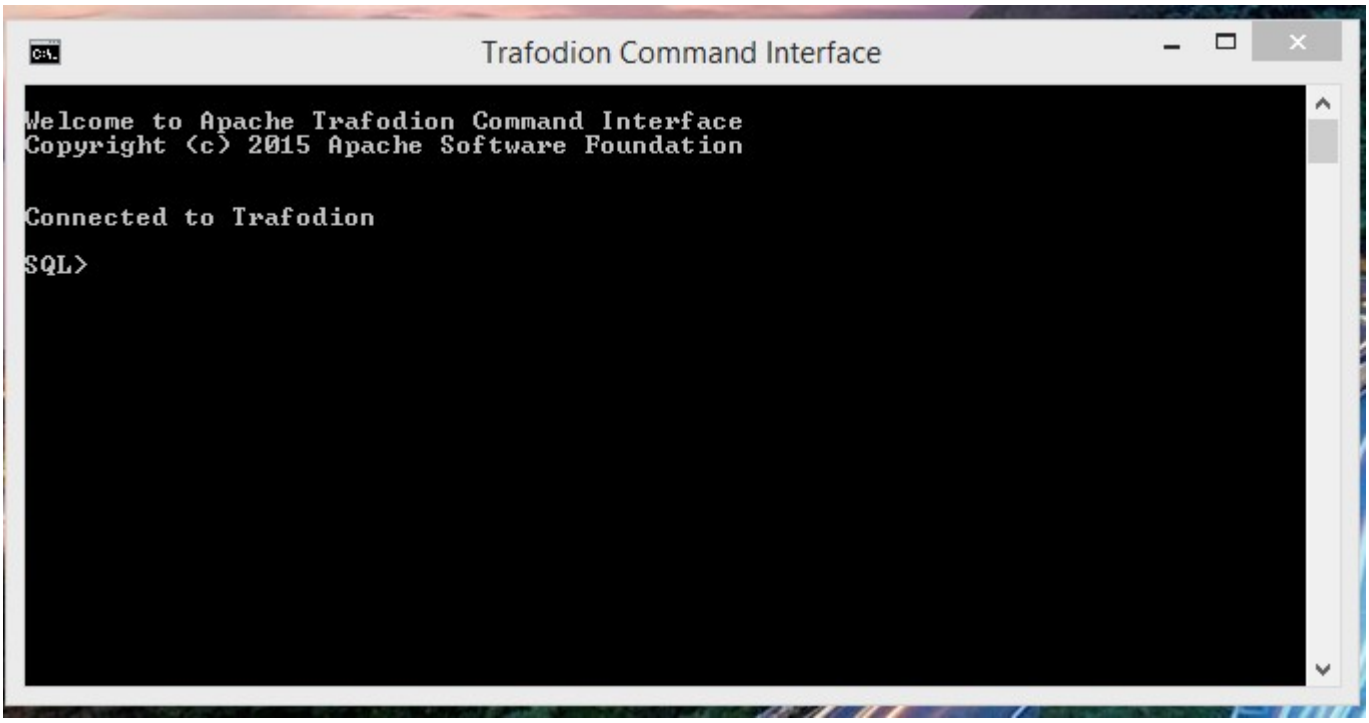
Chapter 5. Run Commands Interactively

After launching TrafCI, you can run SQL statements and interface commands in the command-line interface.

5.1. User Interface

5.1.1. Product Banner

After you launch TrafCI and connect to the database platform, the product banner appears in the command-line interface:



5.1.2. Interface Prompt

The standard prompt is `SQL>`. You can change the prompt, `SQL>`, to something else by using the `SET SQLPROMPT` or `SET PROMPT` command. For more information, Customize the Standard Prompt.

5.1.3. Break the Command Line

You cannot break an interface command over multiple lines. Each interface command must be entered on one line. If you accidentally break an interface command across more than one line, enter the SQL terminator and then reenter the command on one line.

You can continue any SQL statement over multiple lines, breaking that statement at any point except within a word, a numeric literal, or a multi-character operator (for example, <=). To break a string literal in a DML statement, use a concatenation operator (| |). For more information, see the concatenation operator in the *Trafodion SQL Reference Manual*.

To terminate an SQL statement that spans multiple lines, use the SQL terminator for the session. You can also include several SQL statements on the same command line provided that each one is terminated by the SQL terminator. For more information, see Set and Show the SQL Terminator.

5.1.4. Case Sensitivity

In the command-line interface, you can enter SQL statements and interface commands in uppercase, lowercase, or mixed-case characters. All parts of statements and commands are case-insensitive except for parts that you enclose in single-quotes (') or double-quotes (").

5.2. Interface Commands

The interface commands allow you to customize TrafCI (for example, by using SET commands) or return information about the interface settings or database objects (for example, by using SHOW commands).

For more information about the interface commands, see Commands.

Each interface command must be entered on one line. If you accidentally break an interface command across more than one line, enter the SQL terminator and then reenter the command on one line.

5.2.1. Show Session Attributes

To display the attributes and settings of the current TrafCI session, use the ENV, SHOW SESSION, or SESSION command.

Example

This SESSION command displays the session attributes:

```
SQL> SESSION

COLSEP            " "
HISTOPT           DEFAULT [No expansion of script files]
IDLETIMEOUT       0 min(s) [Never Expires]
LIST_COUNT        0 [All Rows]
LOG FILE          c:\session.txt
LOG OPTIONS       APPEND,CMDTEXT ON
MARKUP            RAW
PROMPT            SQL>
SCHEMA            SEABASE
SERVER            sqws135.houston.host.com:37800
SQLTERMINATOR     ;
STATISTICS        OFF
TIME              OFF
TIMING            OFF
USER              user1

SQL>
```

For more information, see ENV Command SHOW SESSION Command.

5.2.2. Set and Show Session Idle Timeout Value

The idle timeout value of a session determines when the session expires after a period of inactivity. To set the idle timeout value of a session, enter the `SET IDLETIMEOUT` command.

Example

This `SET IDLETIMEOUT 0` command sets the idle timeout to an infinite amount of time so that the session never expires:

```
SQL> SET IDLETIMEOUT 0
SQL>
```

To show the idle timeout value that is in effect for the session, enter the `SHOW IDLETIMEOUT` command.

Example

This `SHOW IDLETIMEOUT` command displays an idle timeout of zero minutes, which means that the session never expires:

```
SQL> SHOW IDLETIMEOUT

IDLETIMEOUT 0 min(s) [Never Expires]

SQL>
```

For more information, see the SET IDLETIMEOUT Command and the <<cmd_show_idletimeout, SET IDLETIMEOUT Command>.

5.2.3. Customize the Standard Prompt

To change the standard prompt in the command-line interface, use one or both of these commands:

SET PROMPT Command

The `SET PROMPT` command changes the default prompt to a specified character or string.

Example

This `SET PROMPT` command changes the prompt to the current user (`user1`) and `ENTER>`:

```
SQL>set prompt "%USER ENTER>"

user1 ENTER>
```

For more information, see SET PROMPT Command.

SET TIME Command

The `SET TIME ON` command causes the current time of the client workstation to be displayed in the prompt:

```
SQL ENTER> SET TIME ON

20:32:26 SQL ENTER>
```

The `SET TIME OFF` command removes the current time from the prompt:

```
20:32:26 SQL ENTER> SET TIME OFF

SQL ENTER>
```

For more information, see the SET TIME Command.

5.2.4. Set and Show the SQL Terminator

The SQL terminator symbolizes the end of an SQL statement. By default, the SQL terminator is a semicolon (`;`).

To change the SQL terminator, enter the `SET SQLTERMINATOR` command.

Example

This SET SQLTERMINATOR command sets the SQL terminator to a period (`.`):

```
SQL> SET SQLTERMINATOR .
SQL> INSERT INTO sales.custlist
+> (SELECT * FROM invent.supplier
+> WHERE suppnum=8).

--- 1 row(s) inserted.
SQL>
```

To show the SQL terminator that is in effect for the session, enter the `SHOW SQLTERMINATOR` command.

Example

This `SHOW SQLTERMINATOR` command displays SQLTERMINATOR `.`, where the period (`.`) is the SQL terminator for the session:

```
SQL> SHOW SQLTERMINATOR
SQLTERMINATOR .

SQL>
```

For more information, see the SET SQLTERMINATOR Command and the SHOW SQLTERMINATOR Command.

5.2.5. Display the Elapsed Time

By default, TrafCI does not display the elapsed time of an SQL statement after the statement executes. To display the elapsed time after each SQL statement executes, enter the `SET TIMING ON` command:

```
SQL> SET TIMING ON
SQL> SELECT suppname, street, city, state, postcode
+> FROM invent.supplier
+> WHERE suppnum=3;

SUPPNAME          STREET               CITY           STATE        POSTCODE
----------------- -------------------- -------------- ------------ ----------
HIGH DENSITY INC  7600 EMERSON         NEW YORK       NEW YORK     10230

--- 1 row(s) selected. Elapsed :00:00:00.111 SQL>
```

To prevent the elapsed time from being displayed after each SQL statement executes, enter the `SET TIMING OFF` command:

```
SQL> SET TIMING OFF
SQL> /

SUPPNAME          STREET               CITY           STATE        POSTCODE
----------------- -------------------- -------------- ------------ ----------
HIGH DENSITY INC  7600 EMERSON         NEW YORK       NEW YORK     10230

--- 1 row(s) selected.

SQL>
```

For more information, see the SET TIMING Command.

5.2.6. Set and Show the Current Schema

By default, the schema of the session is USR. The SQL statement, SET SCHEMA, allows you to set the schema for the TrafCI session.

Example

This SET SCHEMA statement changes the default schema to PERSNL for the session:

```
SQL> SET SCHEMA persnl;

--- SQL operation complete.

SQL> DELETE FROM employee
+> WHERE first_name='TIM' AND
+> last_name='WALKER';

--- 1 row(s) deleted.

SQL>
```

The schema that you specify with SET SCHEMA remains in effect until the end of the session or until you execute another SET SCHEMA statement.

If you execute this statement in a script file, it affects not only the SQL statements in the script file but all subsequent SQL statements that are run in the current session. If you set the schema in a script file, reset the default schema for the session at the end of the script file.

For more information about the SET SCHEMA statement, see the *Trafodion SQL Reference Manual*.

The SHOW SCHEMA command displays the current schema for the session.

Example

This SHOW SCHEMA command displays SCHEMA PERSNL, where PERSNL is the name of the current schema for the session:

```
SQL> SHOW SCHEMA SCHEMA persnl
SQL>
```

For more information, SHOW SCHEMA Command.

5.2.7. Limit Query Result Set

To set the maximum number of rows to be returned by `SELECT` statements that are executed in the session, enter the `SET LIST_COUNT` command.

Example

This `SET LIST_COUNT` command limits the result set of queries to 20 rows:

```
SQL> SET LIST_COUNT 20
```

To show the limit that is in effect for the session, enter the `SHOW LIST_COUNT` command.

Example

This `SHOW LIST_COUNT` command shows that the number of rows returned by `SELECT` statements is unlimited:

```
SQL> SHOW LIST_COUNT

LISTCOUNT 0 [All Rows]
```

For more information, see the SET LIST_COUNT Command and SHOW LIST_COUNT Command.

5.2.8. Display Executed Commands

To display commands that were recently executed in the TrafCI session, enter the `HISTORY` command. The `HISTORY` command associates each command with a number that you can use to re-execute or edit the command with the `FC` command. See Edit and Reexecute a Command.

Example

This `HISTORY` command displays a maximum of 100 commands that were entered in the session:

```
SQL> HISTORY

1> SET IDLETIMEOUT 0
2> SET SCHEMA persnl;
3> SELECT * FROM project;

SQL>
```

To save the session history in a user-specified file, enter the `SAVEHIST` command.

Example

This `SAVEHIST` command saves the session history in a file named `history.txt` in the local directory where you are running TrafCI:

```
SQL> SAVEHIST history.txt
```

For more information, see the HISTORY Command and the SAVEHIST Command.

5.2.9. Edit and Reexecute a Command

To edit and reexecute a command in the history buffer of an TrafCI session, enter the FC command. To display the commands in the history buffer, use the HISTORY command. See Display Executed Commands.

Example

This **FC** command and its delete (D) editing command correct a SELECT statement that was entered incorrectly:

```
SQL> FC

SQL> SELECCT FROM employee;
     .... d
SQL> SELECT FROM employee;
```

Pressing Enter executes the corrected SELECT statement. For more information, see the FC Command.

5.2.10. Clear the Interface Window

After entering commands in TrafCI, you can clear the interface window by using the `CLEAR` command.

Example

This `CLEAR` command clears the interface window so that only the prompt appears at the top of the window:

```
SQL> CLEAR
```

For more information, see the CLEAR Command.

5.2.11. Obtain Help

To display help text for an interface command that is supported in TrafCI, enter the `HELP` command.

Example

This `HELP` command displays syntax and examples of the `FC` command:

```
SQL> HELP FC
```

For more information, see the HELP Command.

5.3. Run SQL Statements

In TrafCI, you can run SQL statements interactively. TrafCI supports all the SQL statements, SQL utilities, and other SQL-related commands that the Trafodion database engine supports. For more information about those SQL statements, see the *Trafodion SQL Reference Manual*.

To run SQL statements from script files in TrafCI, see Run Scripts.

5.3.1. Execute an SQL Statement

Example

You can query the `EMPLOYEE` table and return an employee's salary by executing this `SELECT` statement in TrafCI:

```
SQL> SELECT salary
+> FROM persnl.employee
+> WHERE jobcode=100;

SALARY
----------
 175500.00
 137000.10
 139400.00
 138000.40
  75000.00
  90000.00
 118000.00
  80000.00
  70000.00
  90000.00
  56000.00

--- 11 row(s) selected.

SQL>
```

If the SQL statement executes successfully, TrafCI returns a message indicating that the SQL operation was successful, followed by the standard prompt. If a problem occurs during the execution of the SQL statement, TrafCI returns an error message.

5.3.2. Repeat an SQL Statement

To run a previously executed SQL statement, use the `/`, `RUN`, or `REPEAT` command.

```
SQL> /

SALARY
----------
 175500.00
 137000.10
 139400.00
 138000.40
  75000.00
  90000.00
 118000.00
  80000.00
  70000.00
  90000.00
  56000.00

--- 11 row(s) selected.

SQL>
```

For more information, see the / Command, RUN Command, or REPEAT Command.

5.3.3. Prepare and Execute SQL Statements

You can prepare, or compile, an SQL statement by using the PREPARE statement and later execute the prepared SQL statement by using the EXECUTE statement.

Prepare a SQL Statement

Use the PREPARE statement to compile an SQL statement for later execution with the EXECUTE statement. You can also use the PREPARE statement to check the syntax of an SQL statement without executing the statement.

Example

This PREPARE statement compiles a SELECT statement named empsal and detects a syntax error:

```
SQL> PREPARE empsal FROM
+> SELECT salary FROM employee
+> WHERE jobcode = 100;
SQL>
```

You can then correct the syntax of the SQL statement and prepare it again:

```
SQL> PREPARE empsal FROM
+> SELECT salary FROM persnl.employee
+> WHERE jobcode = 100;

--- SQL command prepared.
```

To specify a parameter to be supplied later, either in a `SET PARAM` statement or in the `USING` clause of an `EXECUTE` statement, use one of these types of parameters in the SQL statement:

- Named parameter, which is represented by ?*param-name*
- Unnamed parameter, which is represented by a question mark (?) character

Example

This prepared SELECT statement specifies unnamed parameters for salary and job code:

```
SQL> PREPARE findemp FROM
+> SELECT  FROM persnl.employee
+> WHERE salary > ? AND jobcode = ?;

--- SQL command prepared.
```

This PREPARE statement prepares another SELECT statement named empcom, which has one named parameter, ?*dn*, for the department number, which appears twice in the statement:

```
SQL> PREPARE empcom FROM
+> SELECT first_name, last_name, deptnum
+> FROM persnl.employee
+> WHERE deptnum <> ?dn AND salary <=
+> (SELECT AVG(salary)
+> FROM persnl.employee
+> where deptnum = ?dn);

--- SQL command prepared.
```

For the syntax of the PREPARE statement, see the *Trafodion SQL Reference Manual*.

Setting Parameters

In an TrafCI session, you can set a parameter of an SQL statement (either prepared or not) by using the SET PARAM command.

The parameter name is case-sensitive. If you specify it in lowercase in the SET PARAM command, you must specify it in lowercase in other statements, such as DML statements or EXECUTE.

Example

This SET PARAM command sets a value for the parameter named ?sal, which you can apply to one of the unnamed parameters in the prepared findemp statement or to a named parameter with an identical name in an SQL statement:

```
SQL> SET PARAM ?sal 40000.00
```

This SELECT statement uses sal as a named parameter:

```
SQL> SELECT last_name
+> FROM persnl.employee
+> WHERE salary = ?sal;
```

This SET PARAM command sets a value for the parameter named dn, which you can apply to the named parameter, ?dn, in the prepared empcom statement or to a named parameter with an identical name in an SQL statement:

```
SQL> SET PARAM ?dn 1500
```

For the syntax of the SET PARAM command, see the SET PARAM Command.

To determine what parameters you have set in the current session, use the SHOW PARAM command.

Example

This SHOW PARAM command displays the recent SET PARAM settings:

```
SQL> SHOW PARAM dn 1500
sal 40000.00
SQL>
```

For the syntax of the SHOW PARAM command, SHOW PARAM Command.

Reset the Parameters

To change the value of a parameter, specify the name of the parameter in the RESET PARAM command and then use the SET PARAM command to change the setting.

Example

Suppose that you want to change the salary parameter to 80000.00:

```
SQL> RESET PARAM ?sal
SQL> SET PARAM ?sal 80000.00
SQL>
```

Entering the `RESET PARAM` command without specifying a parameter name clears all parameter settings in the session.

Example

```
SQL> RESET PARAM
SQL> SHOW PARAM
SQL>
```

To use the parameters that you had set before, you must reenter them in the session:

```
SQL> SET PARAM ?dn 1500
SQL> SET PARAM ?sal 80000.00
SQL> SHOW PARAM dn 1500

sal 80000.00

SQL>
```

For the syntax of the `RESET PARAM` command, see the RESET PARAM Command.

5.3.4. Execute a Prepared SQL Statement

To execute a prepared SQL statement, use the EXECUTE statement.

Example

This EXECUTE statement executes the prepared `empsal` statement, which does not have any parameters:

```
SQL> EXECUTE empsal;

SALARY
----------
 137000.10
  90000.00
  75000.00
 138000.40
  56000.00
 136000.00
  80000.00
  70000.00
 175500.00
  90000.00
 118000.00

--- 11 row(s) selected.

SQL>
```

This EXECUTE statement executes the prepared empcom statement, which has one named parameter, ?dn, which was set by SET PARAM for the department number:

```
SQL>EXECUTE empcom;

FIRST_NAME      LAST_NAME            DEPTNUM
--------------- -------------------- -------
ALAN            TERRY                   3000
DAVID           TERRY                   2000
PETE            WELLINGTON              3100
JOHN            CHOU                    3500
MANFRED         CONRAD                  4000
DINAH           CLARK                   9000
DAVE            FISHER                  3200
GEORGE          FRENCHMAN               4000
KARL            HELMSTED                4000
JOHN            JONES                   4000
JOHN            HUGHES                  3200
WALTER          LANCASTER               4000
MARLENE         BONNY                   4000
BILL            WINN                    2000
MIRIAM          KING                    2500
GINNY           FOSTER                  3300
```

```
MARIA           JOSEF                4000
HERB            ALBERT               3300
RICHARD         BARTON               1000
XAVIER          SEDLEMEYER           3300
DONALD          TAYLOR               3100
LARRY           CLARK                1000
JIM             HERMAN               3000
GEORGE          STRICKER             3100
OTTO            SCHNABL              3200
TIM             WALKER               3000
TED             MCDONALD             2000
PETER           SMITH                3300
MARK            FOLEY                4000
HEIDI           WEIGL                3200
ROCKY           LEWIS                2000
SUE             CRAMER               1000
MARTIN          SCHAEFFER            3200
HERBERT         KARAJAN              3200
JESSICA         CRINER               3500

--- 35 row(s) selected.

SQL>
```

This EXECUTE statement executes the prepared findemp statement, which has two unnamed parameters: ?sal, which was set by SET PARAM for the salary, and a parameter that was not set in advance for the job code:

```
SQL> EXECUTE findemp USING ?sal, 100;

EMP_NUM FIRST_NAME      LAST_NAME            DEPTNUM JOBCODE SALARY
------- --------------- -------------------- ------- ------- ---------
    213 ROBERT          WHITE                   1500     100  90000.00
     23 JERRY           HOWARD                  1000     100 137000.10
      1 ROGER           GREEN                   9000     100 175500.00
     29 JANE            RAYMOND                 3000     100 136000.00
     32 THOMAS          RUDLOFF                 2000     100 138000.40
     43 PAUL            WINTER                  3100     100  90000.00
     65 RACHEL          MCKAY                   4000     100 118000.00

--- 7 row(s) selected.
SQL>
```

For the syntax of the EXECUTE statement, see the *Trafodion SQL Reference Manual*.

5.4. Log Output

To log an TrafCI session, use the `SPOOL` or `LOG` command. The `SPOOL` and `LOG` commands record into a log file the commands that you enter in the command-line interface and the output of those commands.

5.4.1. Start the Logging Process

To start logging, enter one of these commands:

- `SPOOL ON` or `LOG ON`
- `SPOOL` *`log-file`* or `LOG` *`log-file`*

For more information, see the LOG Command and the SPOOL Command.

SPOOL ON or LOG ON Command

The SPOOL ON or LOG ON command logs information about a session in the sqlspool.lst file, which TrafCI stores in the bin directory:

- On Windows:

  ```
  <trafci-installation-directory>\Trafodion Command Interface\bin\sqlspool.lst
  ```

  trafci-installation-directory is the directory where you installed the TrafCI software files.

- On Linux:

  ```
  <trafci-installation-directory>/trafci/bin/sqlspool.lst
  ```

  trafci-installation-directory is the directory where you installed the TrafCI software files.

Example

This SPOOL ON command starts logging the session in the sqlspool.lst file:

```
SQL> SPOOL ON
```

5.4.2. SPOOL *log-file* or LOG *log-file* Command

The SPOOL *log-file* and LOG *log-file* commands record information about a session in a log file that you specify. If you specify a directory for the log file, the directory must exist as specified. Otherwise, an error occurs when you try to run the SPOOL or LOG command. If you do not specify a directory for the log file, TrafCI uses the bin directory.

Example

This SPOOL *log-file* command starts logging the session in the persnl_updates.log file in the C:\log directory:

```
SQL> SPOOL C:\log\persnl_updates.log
```

Using the CLEAR Option

The CLEAR option clears the contents of an existing log file before logging new information to the file. If you omit CLEAR, TrafCI appends new information to existing information in the log file.

Example

This SPOOL *log-file* CLEAR command clears existing information from the specified log file and starts logging the session in the log file:

```
SQL> SPOOL C:\log\persnl_updates.log clear
```

Log Concurrent the TrafCI Sessions

If you plan to run two or more TrafCI sessions concurrently on the same workstation, use the `SPOOL` *`log-file`* or `LOG` *`log-file`* command and specify a unique name for each log file. Otherwise, each session writes information to the same log file, making it difficult to determine which information belongs to each session.

5.4.3. Stopping the Logging Process

To stop logging, enter one of these commands:

- `SPOOL OFF`
- `LOG OFF`

Example

This SPOOL OFF command stops logging in an TrafCI session:

```
SQL> SPOOL OFF
```

5.4.4. View the Contents of a Log File

The log file is an ASCII text file that contains all the lines in TrafCI from the time you start logging to the time you stop logging. The logged lines include prompts, entered commands, output from commands, and diagnostic messages.

Example

This log file contains information from when you started logging to when you stopped logging:

```
================================================================================
Spooling started at May 29, 2105 4:52:23 PM
================================================================================

SQL> SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

--- SQL operation complete. SQL>begin work;

--- SQL operation complete.

SQL> DELETE FROM employee WHERE empnum=32;

-- 1 row(s) deleted.

SQL> INSERT INTO employee
(empnum, first_name, last_name, deptnum, salary) values(51, 'JERRY',
'HOWARD', 1000, 137000.00);

-- 1 row(s) inserted.

SQL> UPDATE dept SET manager=50
where deptnum=1000;

--- 1 row(s) updated.

SQL> COMMIT WORK;

--- SQL operation complete.

SQL> LOG OFF
```

Chapter 6. Run Scripts

6.1. Create a Script File

A script file that you run in TrafCI must be an ASCII text file that contains only these elements:

- SQL Statements
- Commands
- Comments
- Section Headers

For an example, see <script_example, Example Script File>>.

You cannot use shell commands in a script file that you run in TrafCI. To create shell scripts that run TrafCI, see Run TrafCI from Perl or Python.

6.2. SQL Statements

Script files support any of the various SQL statements that you can run in TrafCI. For more information about SQL statements, see the *Trafodion SQL Reference Manual*.

6.3. Commands

Most interface commands are supported in script files except the FC command. For a list of the interface commands, see Commands.

6.4. Comments

You can include comments anywhere in a script file. SQL also supports comments. Comments are useful for documenting the functionality of the script file and for debugging. When debugging, use comments to disable specific statements or commands without removing them from the script file.

To denote a comment in a script file, use two hyphens before the comment:

```
-- comment
```

The end of the line marks the end of the comment.

6.5. Section Headers

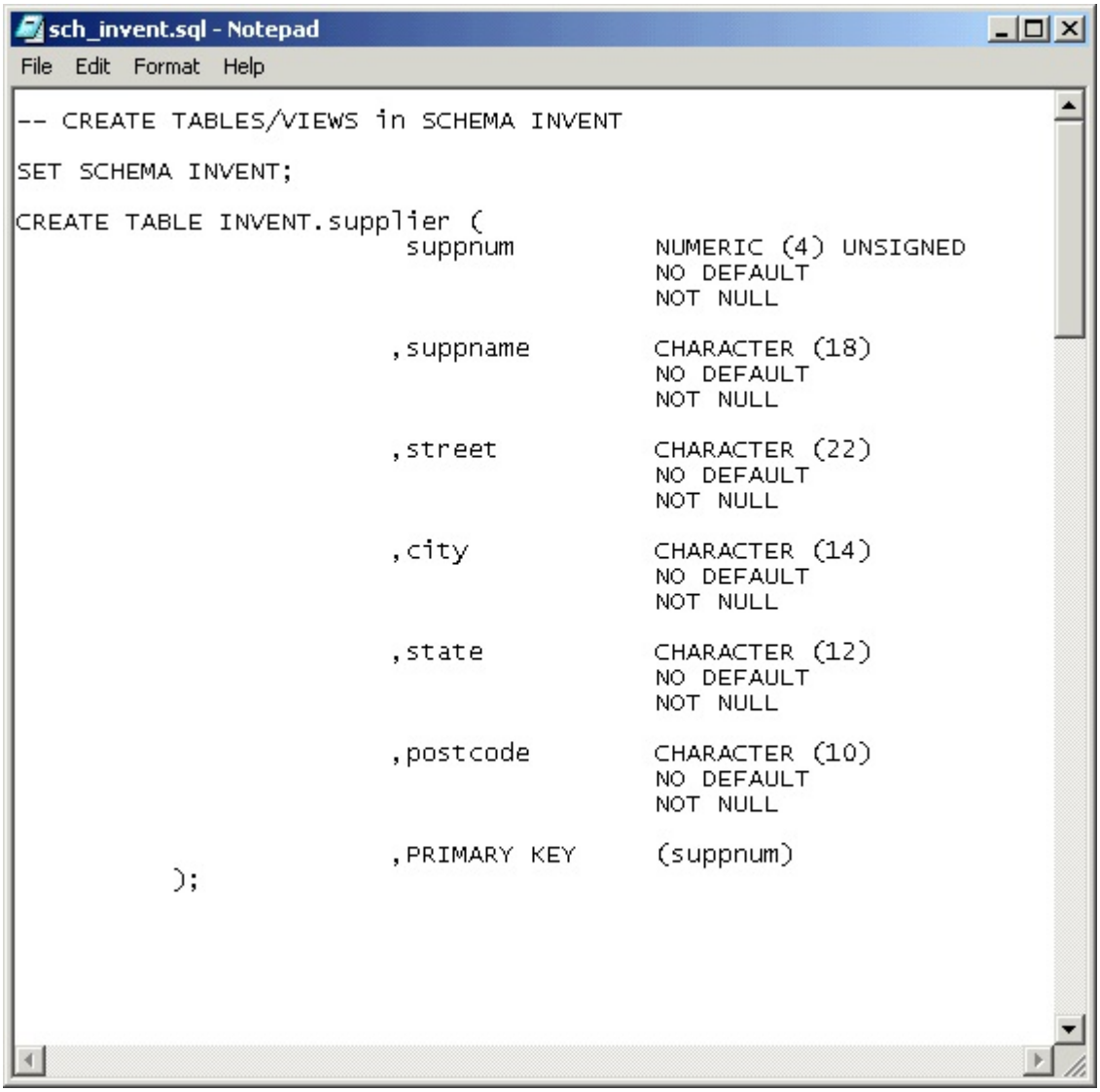
To create sections of commands within a script file, put a section header at the beginning of each section:

```
?SECTION section-name
```

The *section-name* cannot begin with a number or an underscore. Each section name in a script file should be unique because TrafCI executes the first section that it finds that matches the section name in the @ or OBEY command. For more information, see the @ Command OBEY Command.

6.6. Example Script File

This script file creates tables in the inventory schema:



6.7. Run a Script File

To run a script file in TrafCI, use the @ or OBEY command. The @ and OBEY commands run one script file at a time in TrafCI. To run a script file when launching TrafCI, see Run Script When Launching TrafCI.

Example

This @ command runs a script file, `sch_invent.sql`, that creates tables in the inventory schema:

```
@C:\ddl_scripts\sch_invent.sql
```

If the script file is outside the directory of the `trafci.cmd` or `trafci.sh` file (by default, the `bin` directory), you must specify the full path of the script file in the `@` or `OBEY` command.

```
SQL>@C:\ddl_scripts\sch_invent.sql
SQL>-- CREATE SCHEMA
SQL>CREATE SCHEMA INVENT;

--- SQL operation complete.

SQL>-- CREATE TABLES/VIEWS in SCHEMA INVENT
SQL> SET SCHEMA INVENT;

--- SQL operation complete.

SQL>CREATE TABLE INVENT.supplier (
+> suppnum NUMERIC (4) UNSIGNED
+> NO DEFAULT
+> NOT NULL
+> ,suppname CHARACTER (18)
+> NO DEFAULT
+> NOT NULL
+> ,street CHARACTER (22)
+> NO DEFAULT
+> NOT NULL
+> ,city CHARACTER (14)
+> NO DEFAULT
+> NOT NULL
+> ,state CHARACTER (12)
+> NO DEFAULT
+> NOT NULL
+> ,postcode CHARACTER (10)
+> NO DEFAULT
+> NOT NULL
+> ,PRIMARY KEY (suppnum)
+> );

--- SQL operation complete.
```

For more information about the `@` and `OBEY` commands, see the @ Command and the OBEY Command.

6.8. Log Output

To log output of an TrafCI session while running one script file at a time, use the `SPOOL` or `LOG` command. When you run an `OBEY` or `@` command, TrafCI displays each command in the script file, the output for each command, and diagnostic messages in TrafCI. The `SPOOL` or `LOG` command captures this output as it appears in TrafCI and logs it in a log file.

For more information, Log Output.

6.9. Run Scripts in Parallel

In TrafCI, the @ and OBEY commands allow you to run only one script file at a time. However, the PRUN command allows you to run multiple script files simultaneously.

The PRUN command is most useful for running sets of data definition language (DDL) statements simultaneously, which speeds up the process of creating large databases. Put all dependent or related DDL statements in the same script file. For more information on running scripts in parallel using the PRUN command, see the PRUN Command.

Chapter 7. Run TrafCI From Perl or Python

You can execute SQL statements in Perl or Python by invoking the TrafCI Perl or Python wrapper script.

These instructions assume that you installed the TrafCI product. For more information, see Install and Configure.

7.1. Set the Login Environment Variables

Before launching TrafCI from Perl or Python, set these login environment variables:

Environment Variable	Description
TRAFCI_PERL_JSERVER=<JavaServer_jar_path>	Specifies the Perl JavaServer JAR location.
TRAFCI_PYTHON_JSERVER=<Jython_jar_path>	Specifies the Jython JAR file location.
TRAFCI_PERL_JSERVER_PORT=<port_number>	Specifies the port on which the JavaServer is listening.

The Trafodion Command Interface Installer Wizard can attempt to automatically download and install both the Perl JavaServer and Jython open source extensions. If you wish to download and install them manually, refer to the instructions in the `README` file in the samples directory.

To set the login environment variables, see the instructions for the operating system of the client workstation:

- Set the Login Environment Variables on Windows.
- Set the Login Environment Variables on Linux or Unix.

The Perl and Python wrapper scripts do not require these environment variables:

- `TRAFCI_SERVER`
- `TRAFCI_USER`
- `TRAFCI_PASSWORD`

7.1.1. Set the Login Environment Variables on Windows

You can set the login environment variables for the session at command prompts, or you can set the login environment variables for the system or user by including them in the System Properties.

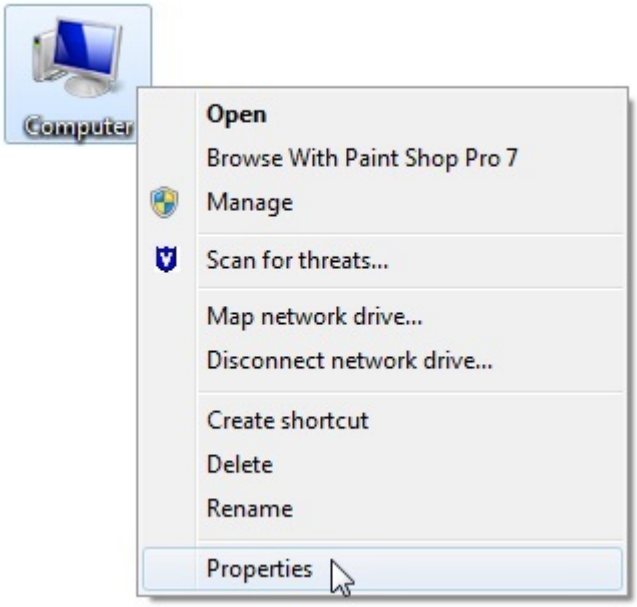
Set Login Environment Variables on the Command Line

At each command prompt, enter one of these commands:

```
set TRAFCI_PERL_JSERVER=<absolute-path-of-JavaServer.jar>
set TRAFCI_PYTHON_JSERVER=<absolute-path-of-Jython.jar>
set TRAFCI_PERL_JSERVER_PORT=<portnumber>
```

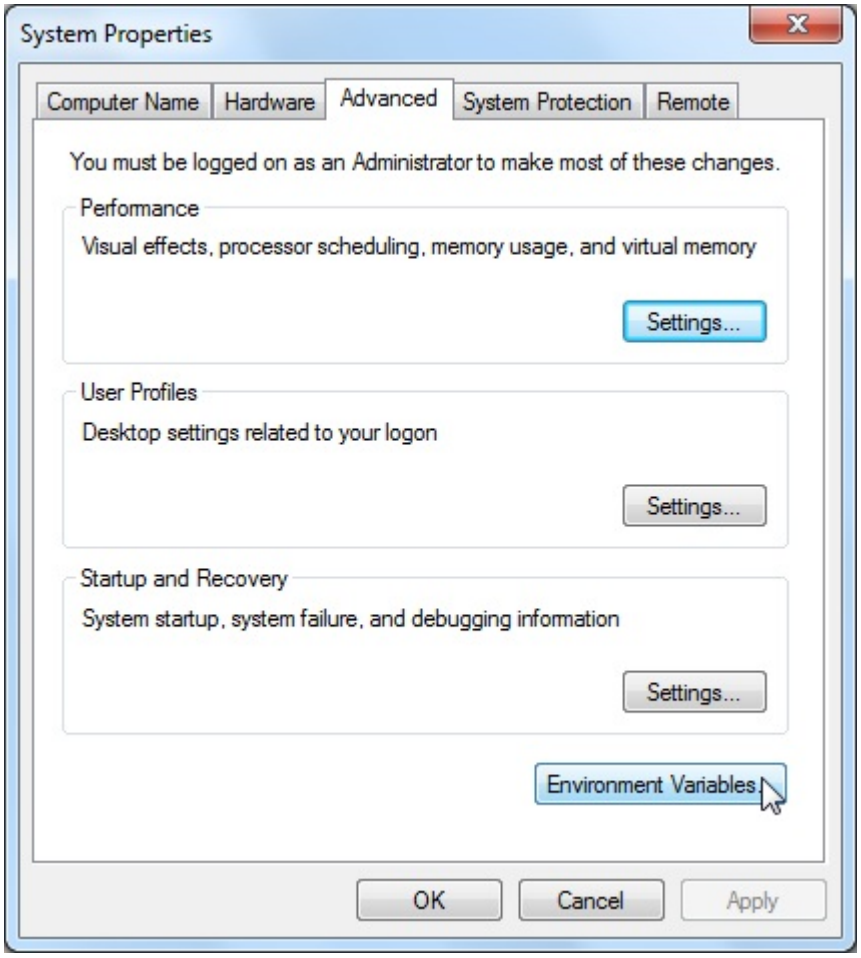
7.1.2. Set Login Environment Variables in the System Properties

1. Right-click the **Computer** icon on your desktop, and then select **Properties**:

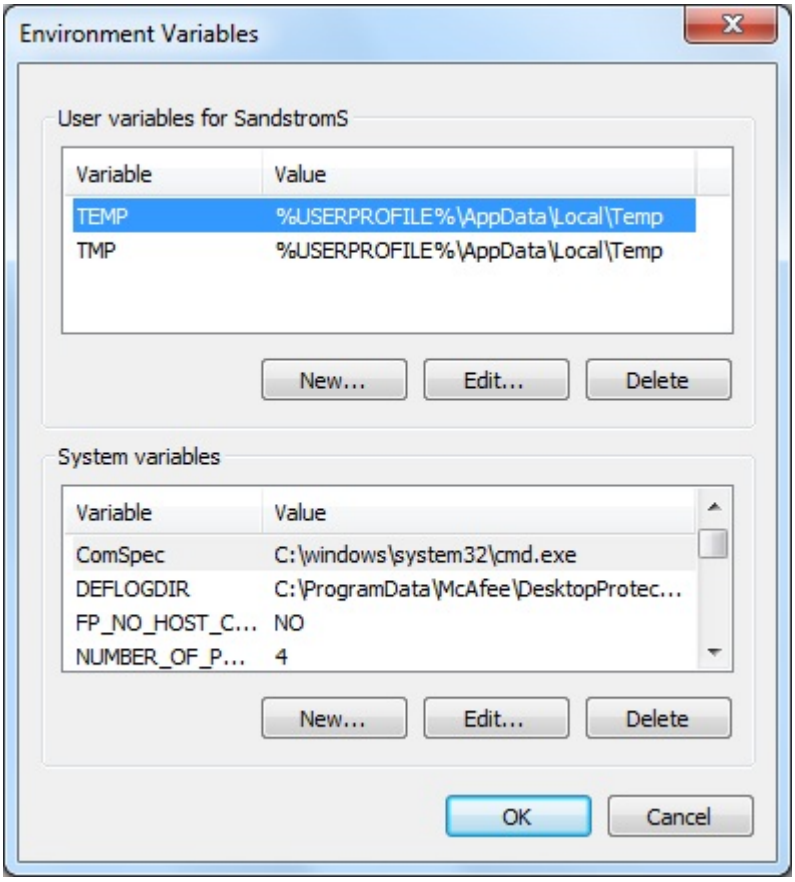


2. In the **Control Panel**, click the **Advanced** system settings.
3. In the **System Properties** dialog box, click the **Advanced** tab.

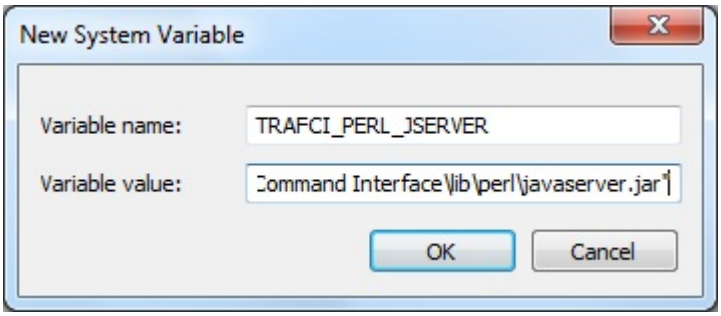
4. Click the **Environment Variables** button:



5. In the **Environment Variables** dialog box, click **New** under *System* or *User* variables, whichever you prefer.



6. In the **New System Variable** (or **New User Variable**) dialog box, type the name of the login environment variable for the *Variable Name* and the required value for the *Variable Value*, and then click **OK**:



7. Verify that the environment variable appears under *System* or *User* variables.
8. Repeat Step 5 to Step 7 for each login environment variable.
9. After adding all three environment variables, click **OK** in the **Environment Variables and System Properties** dialog boxes to accept the changes.

7.1.3. Set the Login Environment Variables on Linux or UNIX

You can set the login environment variables for the session at command prompts, or you can set the login environment variables for each user by including the variables in the user profile on a Linux or UNIX client workstation.

Set Login Environment Variables on the Command Line

At each command prompt in any shell except the C shell, enter one of these commands:

```
export TRAFCI_PERL_JSERVER=<absolute-path-of-JavaServer.jar>
export TRAFCI_PYTHON_JSERVER=<absolute-path-of-Jython.jar>
export TRAFCI_PERL_JSERVER_PORT=<portnumber>
```

At each command prompt in the C shell, enter one of these commands:

```
setenv TRAFCI_PERL_SERVER=<absolute-path-of-JavaServer.jar>
setenv TRAFCI_PYTHON_JSERVER=<absolute-path-of-Jython.jar>
setenv TRAFCI_PERL_JSERVER_PORT=<portnumber>
```

Setting Login Environment Variables in the User Profile

To set the login environment variables in the user profile:

1. Open the user profile (`.profile` or `.bash_profile` for the Bash shell) in the `$HOME` directory.

   Example

   ```
   vi .profile
   ```

2. Add these `export` commands (or `setenv` commands for the C shell) to the user profile.

   Example

   ```
   export TRAFCI_PERL_JSERVER=<absolute-path-of-JavaServer.jar>
   export TRAFCI_PYTHON_JSERVER=<absolute-path-of-Jython.jar>
   export TRAFCI_PERL_JSERVER_PORT=<portnumber>
   ```

3. To activate the changes, either log out and log in again or execute the user profile.

Example

```
. .profile
```

7.2. Perl and Python Wrapper Scripts

The Perl or Python wrapper scripts enable you to run SQL statements and script files using a single connection or multiple connections within Perl or Python programs. The Perl wrapper script is `trafci.pl`, and the Python wrapper script is `trafci.py`. By default, these wrapper scripts are located in the `bin` directory:

Operating System	Directory
Windows	<trafci-installation-directory>\Trafodion Command Interface\bin
Linux/Unix	<trafci-installation-directory>/trafci/bin

trafci-installation-directory is the directory where you installed the TrafCI software files.

7.3. Launch TrafCI From the Perl or Python Command Line

You can launch the Perl or Python wrapper scripts as shown below:

Language	Launch Command	Example
Perl	perl trafci.pl <perl-script-filename>	> perl trafci.pl example.pl
Python	python trafci.py <python-script-filename>	> python trafci.py example.py

7.3.1. Example Perl Program (`sample.pl`)

```
use lib 'C:\\Program Files (x86)\\Apache Software Foundation\\Trafodion Command
Interface\\lib\\perl';
use Session;

# create a new session
$sess = Session->new();

# connect to the database
$sess->connect("user1","password","16.123.456.78","37800");

$retval=$sess->execute(" set schema TRAFODION.CI_SAMPLE ");
print $retval;

# Execute sample queries
$retval=$sess->execute("select * from employee"); print $retval;
$retval=$sess->execute("get statistics"); print $retval;

# disconnect from the database
print "\n\nSession 1: Disconnecting first session. \n\n";
$sess->disconnect();
```

7.3.2. Example Python Program (`sample.py`)

```
import os import sys

## Modify this path
sys.path.append("C:\\Program Files (x86)\\Apache Software Foundation\\Trafodion Command
Interface\\lib\\python")
import Session

# create a new session
sess = Session.Session()

# Connect to the database
x=sess. connect ("user1","password","16.123.456.78","37800")

# Execute sample queries

# execute takes the query string as argument
setSchema = "set schema TRAFODION.CI_SAMPLE"
selectTable = "select * from employee"
getStats = "get statistics"

#Contruct a list of SQL statements to be executed
queryList = [setSchema, selectTable, getStats] print "\n";

for query in queryList:
   print sess.execute (query)

# disconnect the session
sess.disconnect()
del sess
sess=None
```

Chapter 8. Commands

TrafCI supports these commands in the command-line interface or in script files that you run from the command-line interface.

Command	Description	Documentation
@	Runs the SQL statements and interface commands contained in a specified script file.	@ Command
/	Runs the previously executed SQL statement.	/ Command
ALIAS	Maps a string to any interface or SQL command.	ALIAS Command
CLEAR	Clears the command console so that only the prompt appears at the top of the screen.	CLEAR Command
CONNECT	Creates a new connection to the Trafodion database from a current or existing TrafCI session.	CONNECT Command
DELAY	Allows the TrafCI session to be in sleep mode for the specified interval.	DELAY Command
DISCONNECT	Terminates the connection to the Trafodion database.	DISCONNECT Command
ENV	Displays attributes of the current TrafCI session.	ENV Command
EXIT	Disconnects from and exits the command-line interface.	EXIT Command
FC	Edits and re-executes a previous command. This command is restricted to the command-line interface and is disallowed in script files.	FC Command
GET STATISTICS	Returns formatted statistics for the last executed SQL statement.	GET STATISTICS Command
GOTO	Jumps to a point the command history specified by the LABEL Command.	GOTO Command
HELP	Displays help text for the interface commands.	HELP Command
HISTORY	Displays recently executed commands.	HISTORY Command
IF&8230;THEN	Allows the conditional execution of actions specified within the `IF…THEN` conditional statement.	IF…THEN Command
LABEL	Marks a point in the command history that you can jump to by using the GOTO Command.	LABEL Command
LOCALHOST	Executes client machine commands.	LOCALHOST Command
LOG	Logs commands and output from TrafCI to a log file.	LOG Command
OBEY	Runs the SQL statements and interface commands contained in a specified script file.	OBEY Command
PRUN	Runs script files in parallel.	PRUN Command
QUIT	Disconnects from and exits TrafCI.	QUIT Command
RECONNECT	Creates a new connection to the Trafodion database using the login credentials of the last successful connection.	RECONNECT Command
REPEAT	Re-executes a command.	REPEAT Command
RESET LASTERROR	Resets the last error code to `0`.	RESET LASTERROR Command
RESET PARAM	Clears all parameter values or a specified parameter value in the current session.	RESET PARAM Command
RUN	Runs the previously executed SQL statement.	RUN Command
SAVEHIST	Saves the session history in a user-specified file.	SAVEHIST Command

Command	Description	Documentation
SESSION	Displays attributes of the current TrafCI session.	SESSION Command
SET COLSEP	Sets the column separator and allows you to control the formatting of the result displayed for SQL queries.	SET COLSEP Command
SET FETCHSIZE	Changes the default fetchsize used by JDBC.	SET FETCHSIZE Command
SET HISTOPT	Sets the history option and controls how commands are added to the history buffer.	SET HISTOPT Command
SET IDLETIMEOUT	Sets the idle timeout value for the current session.	SET IDLETIMEOUT
SET LIST_COUNT	Sets the maximum number of rows to be returned by SELECT statements that are executed after this command.	SET LIST_COUNT Command
SET MARKUP	Sets the markup format and controls how results are displayed by TrafCI.	SET MARKUP Command
SET PARAM	Sets a parameter value in the current session.	SET PARAM Command
SET PROMPT	Sets the prompt of the current session to a specified string or to a session variable.	SET PROMPT Command
SET SQLPROMPT	Sets the SQL prompt of the current session to a specified string. The default is SQL.	SET SQLPROMPT Command
SET SQLTERMINATOR	Sets the SQL statement terminator of the current session to a specified string. The default is a semicolon (;).	SET SQLTERMINATOR Command
SET STATISTICS	Automatically retrieves the statistics information for a query being executed.	SET STATISTICS Command
SET TIME	Causes the local time of the client workstation to be displayed as part of the interface prompt.	SET TIME Command
SET TIMING	Causes the elapsed time to be displayed after each SQL statement executes.	SET TIMING Command
SHOW ACTIVITYCOUNT	Functions as an alias of SHOW RECCOUNT Command.	SHOW ACTIVITYCOUNT Command
SHOW ALIAS	Displays all or a set of aliases available in the current TrafCI session.	SHOW ALIAS Command
SHOW ALIASES	Displays all the aliases available in the current TrafCI session.	SHOW ALIASES Command
SHOW CATALOG	Displays the current catalog of the TrafCI session.	SHOW CATALOG Command
SHOW COLSEP	Displays the value of the column separator for the current TrafCI session.	SHOW COLSEP Command
SHOW ERRORCODE	Functions as an alias for the SHOW LASTERROR Command.	SHOW ERRORCODE Command
SHOW FETCHSIZE	Displays the fetch size value for the current TrafCI session.	SHOW FETCHSIZE Command
SHOW HISTOPT	Displays the value that has been set for the history option of the current setting.	SHOW HISTOPT Command
SHOW IDLETIMEOUT	Displays the idle timeout value of the current session.	SHOW IDLETIMEOUT Command
SHOW LASTERROR	Displays the last error of the statement that was executed.	SHOW LASTERROR Command
SHOW LIST_COUNT	Displays the maximum number of rows to be returned by SELECT statements in the current session.	SHOW LIST_COUNT Command
SHOW MARKUP	Displays the value that has been set for the markup option for the current TrafCI session.	SHOW MARKUP Command
SHOW PARAM	Displays the parameters that are set in the current session.	SHOW PARAM Command
SHOW PREPARED	Displays the prepared statements in the current TrafCI session.	SHOW PREPARED Command

Command	Description	Documentation
SHOW RECCOUNT	Displays the record count of the previous executed SQL statement.	SHOW RECCOUNT Command
SHOW REMOTEPROCESS	Displays the process name of the DCS server that is handling the current connection.	SHOW REMOTEPROCESS Command
SHOW SCHEMA	Displays the current schema of the TrafCI session.	SHOW SCHEMA Command
SHOW SESSION	Displays attributes of the current TrafCI session.	SHOW SESSION Command
SHOW SQLPROMPT	Displays the value of the SQL prompt for the current session.	SHOW SQLPROMPT Command
SHOW SQLTERMINATOR	Displays the SQL statement terminator of the current session.	SHOW SQLTERMINATOR Command
SHOW STATISTICS	Displays if statistics has been enabled or disabled for the current session.	SHOW STATISTICS Command
SHOW TIME	Displays the setting for the local time in the SQL prompt.	SHOW TIME Command
SHOW TIMING	Displays the setting for the elapsed time.	SHOW TIMING Command
SPOOL	Logs commands and output from TrafCI to a log file.	SPOOL Command
VERSION	Displays the build versions of the platform, database connectivity services, JDBC Type 4 Driver, and TrafCI.	VERSION Command

8.1. @ Command

The @ command executes the SQL statements and interface commands contained in a specified script file. The @ command is executed the same as the OBEY command. For more information on syntax and considerations, OBEY Command.

8.1.1. Syntax

```
@{script-file | wild-card-pattern} [(section-name)]
```

- *script-file*

  is the name of an ASCII text file that contains SQL statements, interface commands, and comments. If the script file exists outside the local directory where you launch TrafCI (by default, the bin directory) specify the full directory path of the script file.

- *wild-card-pattern*

  is a character string used to search for script files with names that match the character string. *wild-card-pattern* matches a string, depending on the operating system for case-sensitivity, unless you enclose it within double quotes. To look for similar values, specify only part of the characters of *wild-card-pattern* combined with these wild-card characters:

*	Use an asterisk () to indicate zero or more characters of any type. For example, *art matches SMART, ARTIFICIAL, and PARTICULAR.
?	Use a question mark (?) to indicate any single character. For example, boo? matches BOOK and BOOT but not BOO or BOOTS.

- (*section-name*)

  is the name of a section within the *script-file* to execute. If you specify *section-name*, the @ command executes the commands between the header line for the specified section and the header line for the next section (or the end of the script file). If you omit *section-name*, the @ command executes the entire script file. For more information, Section Headers.

8.1.2. Considerations

- You must enter the command on one line. The command does not require an SQL terminator.
- Space is disallowed between the @ sign and the first character of the script name.
- For additional considerations, see the OBEY Command.

8.1.3. Examples

- This @ command runs the script file from the local directory (the same directory where you are running TrafCI):

```
SQL> @ddl.sql
```

- This @ command runs the script file in the specified directory on a Windows workstation:

```
 SQL> @c:\my_files\ddl.sql
```

- This @ command runs the script file in the specified directory on a Linux or UNIX workstation:

```
SQL> @./my_files/ddl.sql
```

8.2. / Command

The / command executes the previously executed SQL statement. This command does not repeat an interface command.

8.2.1. Syntax

```
/
```

8.2.2. Considerations

- You must enter the command on one line.
- The command does not require an SQL terminator.

8.2.3. Example

This / command executes the previously executed SELECT statement:

```
SQL> SELECT COUNT() FROM persnl.employee;

(EXPR)
--------------------
62

--- 1 row(s) selected.

`SQL>`/

(EXPR)
--------------------
62

--- 1 row(s) selected.

SQL>
```

8.3. ALIAS Command

The ALIAS command allows you to map a string to any interface or SQL command. The syntax of the interface or SQL command is checked only when the mapped string is executed. This command replaces only the first token of a command string, which allows the rest of the tokens to be treated as parameters.

8.3.1. Syntax

```
ALIAS value AS command SQL-terminator
```

- *value*

  is a case-insensitive string without spaces. *Value* cannot be a command.

- *command*

  is an command or SQL command.

- *SQL-terminator*

  is the default terminator (;) or a string value defined for the statement terminator by the SET SQLTERMINATOR Command. For more information, see Set and Show the SQL Terminator.

8.3.2. Considerations

- You must enter the command on one line. The command does not require an SQL terminator.
- The ALIAS command lasts only for the duration of the session.
- An alias on an alias is not supported.

8.3.3. Examples

- This command creates an alias named `.OS` to perform the `LOCALHOST (LH)` command:

```
SQL> ALIAS .OS AS LH;
```

- This command executes the new `ALIAS` with the `ls` option:

```
SQL> .OS ls

trafci-perl.pl trafci-python.py trafci.cmd trafci.pl trafci.py trafci.sh
```

- This command creates an alias named `.GOTO` to perform the `GOTO` command:

```
SQL> ALIAS .GOTO AS GOTO;
SQL> .GOTO mylabel
```

  The `GOTO` statement executed, ignoring all commands until a `'LABEL MYLABEL'` command is encountered.

- This command creates an alias named USE to perform the `SET SCHEMA` operation, uses the alias to set the schema to `TRAFODION.USR`, and checks the current schema to verify that the alias worked correctly:

```
SQL> ALIAS use AS "SET SCHEMA";
SQL> use TRAFODION.USR;
SQL> SHOW SCHEMA

SCHEMA USR
```

8.4. CLEAR Command

The CLEAR command clears the interface window so that only the prompt appears at the top of the window. CLEAR does not clear the log file or reset the settings of the session.

8.4.1. Syntax

```
CLEAR
```

8.4.2. Considerations

- You must enter the command on one line.
- The CLEAR command does not require an SQL terminator.

8.4.3. Example

This CLEAR command clears the interface window:

```
SQL> CLEAR
```

After the CLEAR command executes, the interface window appears with only the prompt showing:

```
SQL>
```

8.5. CONNECT Command

The `CONNECT` command creates a new connection to the database from the current or existing TrafCI session.

8.5.1. Syntax

```
CONNECT [ username [ /password ][@hostname]]
```

- *`username`*

  specifies the user name for logging in to the database platform.

  - If the user name is not specified, then TrafCI prompts for the user name.
  - If the user name contains spaces or special characters, such as a period (`.`), hyphen (`-`), or underscore (_), then put the name within double quotes. For example: **"sq.user-1"**.

- */password*

  specifies the password of the user for logging in to the database platform.

  - If the password is not specified, then TrafCI prompts for the password.
  - If the password contains spaces or special characters, such as @ or a single quote (`'`), then put the password within double quotes. For example: **"Tr@f0d!0n"**.

- *`@hostname`*

  specifies the host name or IP address of the database platform to which you want the client to connect.

  - If the hostname is not specified, then the value is automatically used from the current TrafCI session.
  - If TrafCI was invoked with the `-noconnect` launch parameter, then you are prompted for a *`hostname`* value.

8.5.2. Considerations

- You must enter the command on one line. The command does not require an SQL terminator.
- If TrafCI was invoked with the `-noconnect` launch parameter, then TrafCI prompts you for the values.
- If the user name or password contains space or special characters, then you must put the name or password within double quotes.

8.5.3. Examples

- This command creates a new connection to the Trafodion database from the current or existing TrafCI session:

```
SQL> CONNECT

User Name: user1
Password:

Connected to Trafodion
```

- This command creates a new connection to the Trafodion database from the current or existing TrafCI session:

```
SQL> CONNECT user1/password

Connected to Trafodion
```

- This command creates a new connection to the Trafodion database from the current or existing TrafCI session:

```
SQL> CONNECT user1/password@host0101

Connected to Trafodion
```

- This command creates a new connection to the Trafodion database from the current or existing TrafCI session:

```
SQL> CONNECT user2

Password:

Connected to Trafodion
```

8.6. DELAY Command

The DELAY command allows the TrafCI session to be in sleep mode for the specified interval.

8.6.1. Syntax

```
DELAY time [sec[ond][s] | min[ute][s]]
```

- *time*

  is an integer.

8.6.2. Considerations

- If seconds or minutes are not specified, then the default is seconds.
- The maximum delay limit is 3600 seconds. You can override this value by setting trafci.maxDelayLimit in _JAVA_OPTIONS. The unit is seconds for trafci.maxDelayLimit.
- This command does not require an SQL terminator.

8.6.3. Examples

- This DELAY command puts the TrafCI session to sleep for 5 seconds before executing the next command:

  ```
  SQL> DELAY 5 secs
  SQL> SHOW VIEWS
  ```

- This DELAY command puts TrafCI session to sleep for 5 minutes before executing the next command, which is to exit the session:

  ```
  SQL> DELAY 5 mins
  SQL> EXIT
  ```

8.7. DISCONNECT Command

The `DISCONNECT` command terminates the connection from the database, not from TrafCI.

8.7.1. Syntax

```
DISCONNECT [WITH] [status] [IF {condition}]
```

- *status*

  is any 1-byte integer. *`status`* is a shell return value, and the range of allowable values is platform dependent.

- *condition*

  is the same as the condition parameter defined for the IF&8230;THEN Command. See Condition Parameter.

8.7.2. Considerations

- You must enter the command on one line. The command does not require an SQL terminator.
- After you disconnect from the Trafodion database, you can still run these interface commands:

ALIAS	HELP	SAVEHIST	SET/SHOW SQLTERMINATOR
CLEAR	HISTORY	SESSION	SET/SHOW TIME
CONNECT	LABEL	SET/SHOW COLSEP	SET/SHOW TIMING
DELAY	LOCALHOST	SET/SHOW HISTOPT	SHOW ALIAS/ALIASES
DISCONNECT	LOG	SET/SHOW IDLETIMEOUT	SHOW SESSION
ENV	QUIT	SET/SHOW MARKUP	SPOOL
EXIT	REPEAT	SET/SHOW PARAM	VERSION
FC	RESET LASTERROR	SET PROMPT	GOTO

8.7.3. Examples

This command terminates the connection to the Trafodion database. You can connect to the Trafodion database by using the `CONNECT` and `RECONNECT` commands:

```
SQL> DISCONNECT

Session Disconnected. Please connect to the database by using
connect/reconnect command.
```

8.8. ENV Command

ENV displays attributes of the current TrafCI session. You can also use the SESSION and SHOW SESSION commands to perform the same function.

8.8.1. Syntax

```
ENV
```

8.8.2. Considerations

- You must enter the command on one line. The command does not require an SQL terminator.
- ENV displays these attributes:

Attribute	Description
COLSEP	Current column separator, which is used to control how query results are displayed. For more information, see SET COLSEP Command.
HISTOPT	Current history options, which controls how the commands are added to the history buffer. For more information, see SET HISTOPT Command.
IDLETIMEOUT	Current idle timeout value, which determines when the session expires after a period of inactivity. By default, the idle timeout is 30 minutes. For more information, see Set and Show Session Idle Timeout Value and SET IDLETIMEOUT Command.
LIST_COUNT	Current list count, which is the maximum number of rows that can be returned by SELECT statements. By default, the list count is all rows. For more information, see SET LIST_COUNT Command.
LOG FILE	Current log file and the directory containing the log file. By default, logging during a session is turned off. For more information, see Log Output, and LOG Command or SPOOL Command.
LOG OPTIONS	Current logging options. By default, logging during a session is turned off, and this attribute does not appear in the output. For more information, see the LOG Command or SPOOL Command.
MARKUP	Current markup option selected for the session. The default option is RAW. For more information, SET MARKUP Command.
PROMPT	Current prompt for the session. For example, the default is SQL>. For more information, Customize the Standard Prompt and SET PROMPT Command.
SCHEMA	Current schema. The default is USR. For more information, see Set and Show the Current Schema.
SERVER	Host name and port number that you entered when logging in to the database platform. For more information, see Log In to Database Platform.
SQLTERMINATOR	Current SQL statement terminator. The default is a semicolon (;). For more information, see Set and Show the SQL Terminator and SHOW SQLTERMINATOR Command.
STATISTICS	Current setting (on or off) of statistics. For more information, see the SET STATISTICS Command.
TIME	Current setting (on or off) of the local time as part of the prompt. When this command is set to on, military time is displayed. By default, the local time is off. For more information, see Customize the Standard Prompt and SET TIME Command.
TIMING	Current setting (on or off) of the elapsed time. By default, the elapsed time is off. For more information, see Display the Elapsed Time and SET TIMING Command.

Attribute	Description
USER	User name that you entered when logging in to the database platform. For more information, Log In to Database Platform.

8.8.3. Examples

- This ENV command displays the attributes of the current session:

```
SQL> ENV

COLSEP            " "
HISTOPT           DEFAULT [No expansion of script files]
IDLETIMEOUT       0 min(s) [Never Expires]
LIST_COUNT        0 [All Rows]
LOG FILE          c:\session.txt
LOG OPTIONS       APPEND,CMDTEXT ON
MARKUP            RAW
PROMPT            SQL>
SCHEMA            SEABASE
SERVER            sqws135.houston.host.com:37800
SQLTERMINATOR     ;
STATISTICS        OFF
TIME              OFF
TIMING            OFF
USER              user1
```

- This ENV command shows the effect of setting various session attributes:

```
4:16:43 PM > ENV

COLSEP            " "
HISTOPT           DEFAULT [No expansion of script files]
IDLETIMEOUT       30 min(s)
LIST_COUNT        0 [All Rows]
LOG               OFF
MARKUP            RAW
PROMPT            SQL>
SCHEMA            SEABASE
SERVER            sqws135.houston.host.com:37800
SQLTERMINATOR     ;
STATISTICS        OFF
TIME              OFF
TIMING            OFF
USER              user1

4:16:49 PM >
```

8.9. EXIT Command

The `EXIT` command disconnects from and exits TrafCI. `EXIT` can return a status code. If no status code is specified, then `0` (zero) is returned by default. In addition, a conditional statement can be appended to the command.

8.9.1. Syntax

```
EXIT [WITH] [status] [IF {condition}]
```

- *`status`*

  is any 1-byte integer. *`status`* is a shell return value, and the range of allowable values is platform dependent.

- *`condition`*

  is the same as the condition parameter defined for the IF&8230;THEN Command. See Condition Parameter.

8.9.2. Considerations

You must enter the command on one line. The command does not require an SQL terminator.

8.9.3. Examples

- This command disconnects from and exits TrafCI, which disappears from the screen:

  ```
  SQL> EXIT
  ```

- In a script file, the conditional exit command causes the script file to quit running and disconnect from and exit TrafCI when the previously run command returns error code `4082`:

```
LOG c:\errorCode.log
SELECT * FROM employee;
EXIT IF errorcode=4082
LOG OFF
```

  These results are logged when error code 4082 occurs:

```
SQL> SELECT * FROM employee;

**** ERROR[4082] Table, view or stored procedure TRAFODION.USR.EMPLOYEE does not
exist or is inaccessible.

SQL> EXIT IF errorcode=4082
```

- The following two examples are equivalent:

```
SQL> EXIT -1 IF LASTERROR <> 0
SQL> EXIT WITH -1 IF LASTERROR != 0
```

- This example exits TrafCI if the last error code is equal to `4082`:

```
SQL> EXIT WITH 82 IF LASTERROR == 4082
SQL> EXIT -- default status is 0
```

8.10. FC Command

The FC command allows you to edit and reissue a command in the history buffer of an TrafCI session. You can display the commands in the history buffer by using the HISTORY command. For information about the history buffer, see the HISTORY Command.

8.10.1. Syntax

```
FC [text | [-]number]
```

- *text*

  is the beginning text of a command in the history buffer. Case is not significant in matching the text to a command.

- *number*

  is either a positive integer that is the ordinal number of a command in the history buffer or a negative integer that indicates the position of a command relative to the most recent command.

Without text or number, FC retrieves the most recent command.

8.10.2. Considerations

- You must enter the command on one line. The command does not require an SQL terminator.
- You cannot execute this command in a script file. You can execute this command only at a command prompt.
- As each line of the command is displayed, you can modify the line by entering these editing commands (in uppercase or lowercase letters) on the line below the displayed command line:

Edit Command	Description
`D`	Deletes the character immediately above the letter `D`. Repeat to delete more characters.
`I`*characters*	Inserts characters in front of the character immediately above the letter `I`.
`R`*characters*	Replaces existing characters one-for-one with characters, beginning with the character immediately above the letter `R`.
characters	Replaces existing characters one-for-one with characters, beginning with the first character immediately above characters. `*characters*` must begin with a non-blank character.

To specify more than one editing command on a line, separate the editing commands with a double slash (`//`). The end of a line terminates an editing command or a set of editing commands.

After you edit a line of the command, TrafCI displays the line again and allows you to edit it again. Press **Enter** without specifying editing commands to stop editing the line. If that line is the last line of the command, pressing **Enter** executes the command.

To terminate a command without saving changes to the command, use the double slash (`//`), and then press **Enter**.

8.10.3. Examples

- Re-execute the most recent command that begins with SH:

```
SQL> FC SH
SQL> SHOW SCHEMA
....
```

Pressing **Enter** executes the `SHOW SCHEMA` command and displays the current schema, `PERSNL`:

```
SQL> FC SH
SQL> SHOW SCHEMA
....

SCHEMA PERSNL

SQL>
```

- Correct an SQL statement that you entered incorrectly by using the delete (D) editing command:

```
SQL> SELECT * FROM persnl.employee;

*** ERROR[15001] A syntax error occurred at or before:
SELECCT * FROM persnl.employee;
      ^
SQL> FC
SQL> SELECCT * FROM persnl.employee;
....     d
SQL>SELECT * FROM persnl.employee;
....
```

Pressing **Enter** executes the corrected SELECT statement.

- Correct an SQL statement that you entered incorrectly by using more than one editing command:

```
SQL> SELT * FROMM persnl.employee;

*** ERROR[15001] A syntax error occurred at or before:
SELT * FROMM persnl.employee;
   ^
SQL> FC
SQL> SELT *  FROMM persnl.employee;
....    iEX//   d
SQL> SELECT * FROM persnl.employee;
....
```

Pressing **Enter** executes the corrected SELECT statement.

- Modify a previously executed statement by replacing a value in the WHERE clause with another value:

```
SQL> SELECT first_name, last_name
+> FROM persnl.employee
+> WHERE jobcode=111;

--- 0 row(s) selected.

SQL> FC
SQL> SELECT first_name, last_name
....
SQL> FROM persnl.employee
....
SQL> WHERE jobcode=111;
                   450
....
SQL> WHERE jobcode=450;
....
```

Pressing Enter lists the first and last names of all of the employees whose job code is 450.

- Modify a previously executed statement by replacing a column name in the select list with another column name:

```
SQL> SELECT first_name, last_name
+> FROM persnl.employee
+> WHERE jobcode=450;

FIRST_NAME      LAST_NAME
--------------- --------------------
MANFRED         CONRAD
WALTER          LANCASTER
JOHN            JONES
KARL            HELMSTED
THOMAS          SPINNER

--- 5 row(s) selected.

SQL> FC
SQL> SELECT first_name, last_name
....        R   empnum,
SQL> SELECT     empnum, last_name
....

SQL> FROM persnl.employee
....

SQL> WHERE jobcode=450;
....
```

Pressing **Enter** lists the employee number and last names of all employees whose job code is `450`:

```
EMPNUM LAST_NAME
------ --------------------
   180 CONRAD
   215 LANCASTER
   216 JONES
   225 HELMSTED
   232 SPINNER

--- 5 row(s) selected.
SQL>
```

8.11. GET STATISTICS Command

The GET STATISTICS command returns formatted statistics for the last executed SQL statement.

8.11.1. Syntax

```
GET STATISTICS
```

8.11.2. Description of Returned Values

Value	Description
Records Accessed	Number of rows returned by disk process to `EID` (Executor In Disk process).
Records Used	Number of rows returned by `EID` after selection.
Disk IOs	Number of actual disk IOs done by disk process.
Message Count	Number of messages sent/received between file system and disk process.
Message Bytes	Number of message bytes sent/received between file system and disk process.
Lock Escl	Number of lock escalations.
Lock Wait	Number of lock waits.
Disk Process Busy Time	CPU time for disk process processes for the specified table.

8.11.3. Considerations

The command requires an SQL terminator.

8.11.4. Examples

```
SQL> SELECT * FROM job;

JOBCODE JOBDESC
------- ------------------
100     MANAGER
1234    ENGINEER
450     PROGRAMMER
900     SECRETARY
300     SALESREP
500     ACCOUNTANT
400     SYSTEM ANALYST
250     ASSEMBLER
420     ENGINEER
600     ADMINISTRATOR
200     PRODUCTION SUPV

--- 11 row(s) selected.

SQL> GET STATISTICS;

Start Time         21:45:34.082329
End Time           21:45:34.300265
Elapsed Time       00:00:00.217936
Compile Time       00:00:00.002423
Execution Time     00:00:00.218750


Table Name         Records  Records Disk Message Message Lock   Lock Disk Process
                   Accessed Used    I/Os Count   Bytes   Escl   Wait Busy Time
TRAFODION.TOI.JOB         2            2       0       4 15232     0    0     363

--- SQL operation complete.
```

8.12. GOTO Command

The GOTO command allows you to jump to a designated point in the command history. The point in the command history is designated by a `LABEL` command. All commands executed after a `GOTO` statement are ignored until the specified label is set. To set a label, use the LABEL Command.

8.12.1. Syntax

```
GOTO {label}
```

- *`label`*

  is a string of characters without quotes and spaces, or a quoted string.

8.12.2. Considerations

- You must enter the command on one line.
- The `GOTO` command cannot currently jump back in the command history; it is a forward-only command.

8.12.3. Examples

These examples show the use of the `GOTO` and `LABEL` commands:

```
SQL> GOTO ViewManagers
SQL> SELECT  FROM Employees; -- skipped
SQL> SHOW RECCOUNT;          -- skipped
SQL> LABEL ViewManagers
SQL> SELECT  FROM Managers;
SQL> GOTO "View Customers"
SQL> SELECT  FROM Invoices;  -- skipped
SQL> LABEL "View Customers"
SQL> SELECT  FROM Customers;
```

8.13. HELP Command

The HELP command displays help text for the commands. See Commands for a descriptions of the commands.

8.14. Syntax

```
HELP [command-name]
```

command-name

is the name of a command.

- If you do not specify a command, then TrafCI returns a list of all commands.
- If you specify SET, then TrafCI returns a list of all SET commands.
- If you specify SHOW, then TrafCI returns a list of all SHOW commands.

8.14.1. Considerations

You must enter the command on one line. The command does not require an SQL terminator.

8.14.2. Examples

- This `HELP` command lists all the interface commands that are supported:

```
SQL> HELP
```

- This `HELP` command lists all the `SET` commands that are supported:

```
SQL> HELP SET
```

- This `HELP` command lists all the `SHOW` commands that are supported:

```
SQL> HELP SHOW
```

- This `HELP` command shows help text for `SET IDLETIMEOUT`:

```
SQL> HELP SET IDLETIMEOUT
```

8.15. HISTORY Command

The HISTORY command displays recently executed commands, identifying each command by a number that you can use to re-execute or edit the command.

8.15.1. Syntax

```
HISTORY [number]
```

- *number*

  is the number of commands to display. The default number is 10. The maximum number is 100.

8.15.2. Considerations

- You must enter the command on one line. The command does not require an SQL terminator.
- You can use the FC command to edit and re-execute a command in the history buffer, or use the REPEAT command to re-execute a command without modifying it. See FC Command or REPEAT Command.

8.15.3. Example

Display the three most recent commands and use FC to redisplay one:

```
SQL> HISTORY 3

14> SET SCHEMA SALES;
15> SHOW TABLES
16> SHOW VIEWS

SQL> FC 14

SQL> SET SCHEMA sales
....
```

Now you can use the edit capabilities of FC to modify and execute a different SET SCHEMA statement.

8.16. IF…THEN Command

`IF…THEN` statements allow for the conditional execution of actions. If the condition is met, the action is executed; otherwise, no action is taken.

8.16.1. Syntax

```
IF {condition} THEN {action} {SQL-terminator}
```

- *`condition`*

  The condition parameter (*`condition`*) is a Boolean statement structured as follows:

  ```
  ( {variable-name | value} {operator} {variable-name | value}
  ```

- *`variable-name`*

  is one of:

  ```
  { LASTERROR
  | RECCOUNT
  | ACTIVITYCOUNT
  | ERRORCODE
  | [%]any ENV variable | any SQL parameter
  }
  ```

- *`value`*

  is any integer or a quoted string, where the quoted string is any non-quote character. \ is the optional escape character.

- *operator*

  is one of:

Operator	Meaning
== | =	equal to
<> | != | ~= | ^=	not equal to
>	greater than
>=	greater than or equal to
<	less than
<=	less than or equal to

- *action*

  The action parameter (*action*) is any interface or SQL command.

- *SQL Terminator*

  The SQL terminator (*SQL-terminator*) is the default terminator (;) or a string value defined for the statement terminator by the SET SQLTERMINATOR Command. See Set and Show the SQL Terminator.

8.16.2. Considerations

- IF…THEN is itself an action. Thus, nested IF…THEN statements are allowed.
- An action must end with the SQL terminator, even if the action is an interface command.

8.16.3. Examples

These commands show multiple examples of `IF…THEN` statements:

```
SQL> INVOKE employees
SQL> -- ERROR 4082 means the table does not exist
SQL> IF ERRORCODE != 4082 THEN GOTO BeginPrepare
SQL> CREATE TABLE employees(ssn INT PRIMARY KEY NOT NULL NOT DROPPABLE, fname
VARCHAR(50), lname VARCHAR(50), hiredate DATE DEFAULT CURRENT_DATE);
SQL> LABEL beginprepare
SQL> PREPARE empSelect FROM
+> SELECT * FROM
+> employees
+> WHERE SSN=?empssn;
SQL> IF user == "alice" THEN SET PARAM ?empssn 987654321;
SQL> IF %user == "bob" THEN SET PARAM ?empssn 123456789;
SQL> EXECUTE empselect
SQL> IF user == "alice" THEN
+> IF activitycount == 0 THEN GOTO insertalice;
SQL> IF user == "bob" THEN IF activitycount == 0 THEN GOTO insertbob;
SQL> EXIT
SQL> LABEL insertalice
SQL> INSERT INTO employees(ssn, fname, lname) VALUES(987654321, 'Alice', 'Smith');
SQL> EXIT
SQL> LABEL insertbob
SQL> INSERT INTO employees(ssn, fname, lname) VALUES(123456789, 'Bob', 'Smith');
SQL> EXIT
```

8.17. LABEL Command

The LABEL command marks a point in the command history that you can jump to by using the `GOTO` command. For more information, see the GOTO Command.

8.17.1. Syntax

```
LABEL {label}
```

- *`label`*

  is a string of characters without quotes and spaces, or a quoted string.

8.17.2. Considerations

You must enter the command on one line.

8.17.3. Examples

- This command creates a label using a string of characters:

  ```
  SQL> LABEL MyNewLabel
  ```

- This command creates a label using a quoted string:

  ```
  SQL> LABEL "Trafodion Label"
  ```

8.18. LOCALHOST Command

The LOCALHOST command allows you to execute client machine commands.

8.18.1. Syntax

```
LOCALHOST | LH <client M/C commands>
```

8.18.2. Considerations

- You must enter the command on one line. The command does not require an SQL terminator.
- The LOCALHOST command has a limitation. When input is entered for the operating system commands (for example, date, time, and cmd), the input is not visible until you hit the enter key.
- If the SET TIMING is set to ON, the elapsed time information is displayed.

8.18.3. Examples

- If you are using a Windows system, dir lists the contents of the directory name. Similarly, if you are on a UNIX system you enter LOCALHOST LS to display the contents of the folder.

```
SQL> LOCALHOST dir

Volume in drive C is E-Client
Volume Serial Number is DC4F-5B3B

Directory of c:\Program Files (x86)\Apache Software Foundation\Trafodion Command

Interface\bin 05/11/2105 01:17 PM <DIR>
05/11/2105 01:17 PM <DIR>
05/16/2105 09:47 AM      1,042 trafci-perl.pl
05/16/2105 09:47 AM      1,017 trafci-python.pl
05/16/2105 09:47 AM        752 trafci.cmd
05/16/2105 09:47 AM      1,416 trafci.pl
05/16/2105 09:47 AM      2,388 trafci.py
05/16/2105 09:47 AM      3,003 trafci.sh
          6 Files(s) 19,491 bytes
          2 Dir (s) 57,686,646,784 bytes free

SQL> LH mkdir c:\trafci -- Will create a directory c:\trafci on your local machine.
```

- This command displays the elapsed time information because the SET TIMING command is set to ON:

```
SQL> SET TIMING ON
SQL> LOCALHOST ls

trafci-perl.pl
trafci-python.py
trafci.cmd
trafci.pl
trafci.py
trafci.sh

Elapsed :00:00:00.078
```

8.19. LOG Command

The LOG command logs the entered commands and their output from TrafCI to a log file. If this is an obey script file, then the command text from the obey script file is shown on the console.

8.19.1. Syntax

```
LOG { ON [CLEAR, QUIET, CMDTEXT {ON | OFF}]
    | log-file [CLEAR, QUIET, CMDTEXT {ON | OFF}]
    | OFF
    }
```

- ON

  starts the logging process and records information in the sqlspool.lst file in the bin directory.

- CLEAR

  instructs TrafCI to clear the contents of the sqlspool.lst file before logging new information to the file.

- QUIET

  specifies that the command text is displayed on the screen, but the results of the command are written only to the log file and not to the screen.

- CMDTEXT ON

  specifies that the command text and the log header are displayed in the log file.

- CMDTEXT OFF

  specifies that the command text and the log header are not displayed in the log file.

- *log-file*

  is the name of a log file into which TrafCI records the entered commands and their output. If you want the log file to exist outside the local directory where you launch TrafCI (by default, the bin directory), specify the full directory path of the log file. The log file does not need to exist, but the specified directory must exist before you execute the LOG command.

- *log-file* CLEAR

  instructs TrafCI to clear the contents of the specified *log-file* before logging new information to the file.

- OFF

  stops the logging process.

8.19.2. Considerations

- You must enter the command on one line. The command does not require an SQL terminator.
- Use a unique name for each log file to avoid writing information from different TrafCI sessions into the same log file.

8.19.3. Examples

- This command starts the logging process and records information to the `sqlspool.lst` file in the `bin` directory:

```
SQL> LOG ON
```

- This command starts the logging process and appends new information to an existing log file, `persnl_updates.log`, in the local directory (the same directory where you are running TrafCI):

```
SQL> LOG persnl_updates.log
```

- This command starts the logging process and appends new information to a log file, `sales_updates.log`, in the specified directory on a Windows workstation:

```
SQL> LOG c:\log_files\sales_updates.log
```

- This command starts the logging process and appends new information to a log file, `sales_updates.log`, in the specified directory on a Linux or UNIX workstation:

```
SQL> LOG ./log_files/sales_updates.log
```

- This command starts the logging process and clears existing information from the log file before logging new information to the file:

```
SQL> LOG persnl_ddl.log CLEAR
```

- This command start the logging process, clears existing information from the log file, and specifies that the command text and log header is not displayed in the log file:

```
SQL> LOG c:\temp\a.txt clear, CMDTEXT OFF
SQL> (SELECT * FROM trafodion.toi.job
+>;

JOBCODE JOBDESC
------- ------------------
100     MANAGER
450     PROGRAMMER 900 SECRETARY
300     SALESREP
500     ACCOUNTANT
400     SYSTEM ANALYST
250     ASSEMBLER
420     ENGINEER
600     ADMINISTRATOR
200     PRODUCTION SUPV

--- 10 row(s) selected.

SQL> log off

Output of c:\temp\a.txt

JOBCODE JOBDESC
------- ------------------
100     MANAGER
450     PROGRAMMER 900 SECRETARY
300     SALESREP
500     ACCOUNTANT
400     SYSTEM ANALYST
250     ASSEMBLER
420     ENGINEER
600     ADMINISTRATOR
200     PRODUCTION SUPV

--- 10 row(s) selected
```

- This command start the logging process, clears existing information from the log file, specifies that no output appears on the console window, and the quiet option is enabled:

```
SQL> LOG c:\temp\b.txt CLEAR, CMDTEXT OFF, QUIET
SQL> SELECT
+> FROM trafodion.toi.job; +
SQL> LOG OFF

Output of c:\temp\b.txt

JOBCODE JOBDESC
------- ------------------
100     MANAGER
450     PROGRAMMER 900 SECRETARY
300     SALESREP
500     ACCOUNTANT
400     SYSTEM ANALYST
250     ASSEMBLER
420     ENGINEER
600     ADMINISTRATOR
200     PRODUCTION SUPV

--- 10 row(s) selected
```

  This command stops the logging process:

```
SQL> LOG OFF
```

For more information, see Log Output.

8.20. OBEY Command

The OBEY command executes the SQL statements and interface commands of a specified script file or an entire directory. This command accepts a single filename or a filename with a wild-card pattern specified. Executing the OBEY command without optional parameters prompts you to enter a filename. If a filename is not specified, then `*.sql` is used.

8.20.1. Syntax

```
OBEY {script-file | wild-card-pattern} [(section-name)]
```

- *script-file*

  is the name of an ASCII text file that contains SQL statements, interface commands, and comments. If the script file exists outside the local directory where you launch TrafCI (by default, the `bin` directory), specify the full directory path of the script file.

- *wild-card-pattern*

  is a character string used to search for script files with names that match the character string. *wild-card-pattern* matches a string, depending on the operating system for case-sensitivity, unless you enclose it within double quotes. To look for similar values, specify only part of the characters of *wild-card-pattern* combined with these wild-card characters:

- (*section-name*)

  is the name of a section within the *script-file* to execute. If you specify *section-name*, the OBEY command executes the commands between the header line for the specified section and the header line for the next section (or the end of the script file). If you omit *section-name*, the OBEY command executes the entire script file. For more information, see Section Headers.

8.20.2. Considerations

- You must enter the command on one line. The command does not require an SQL terminator.
- Put a space between `OBEY` and the first character of the file name.
- You can execute this command in a script file.
- Before putting dependent SQL statements across multiple files, consider the order of the file execution. If a directory is not passed to the `OBEY` command, the file or wild card is assumed to be in the current working directory.
- If the (`*`) is issued in the `OBEY` command, all files are executed in the current directory. Some of the files in the directory could be binary files. The `OBEY` command tries to read those binary files and junk or invalid characters are displayed on the console. For example, this command causes invalid characters to be displayed on the console:

```
SQL> OBEY C:\trafci\bin\
```

- `OBEY` detects recursive obey files (for example, an SQL file that calls OBEY on itself) and prevents infinite loops using a max depth environment variable. If no variable is passed to the JVM, the default depth is set to `10`. To change this depth (for example to a value of `20`), pass a Java environment variable as follows:

```
-Dtrafci.obeydepth=20
```

8.20.3. Examples

- This OBEY command runs the script file from the local directory (the same directory where you are running TrafCI):

```
SQL> OBEY ddl.sql
```

- This OBEY command runs the script file in the specified directory on Windows.

```
SQL> OBEY c:\my_files\ddl.sql
```

- This OBEY command runs the script file in the specified directory on a Linux or UNIX workstation:

```
SQL> OBEY ./my_files/ddl.sql
```

- This sample file contains sections to be used in conjunction with the OBEY command:

```
?section droptable
DROP TABLE course ;

?section create
CREATE TABLE course ( cno VARCHAR(3) NOT NULL
                    , cname VARCHAR(22) NOT NULL
                    , cdescp VARCHAR(25) NOT NULL
                    , cred INT
                    , clabfee NUMERIC(5,2)
                    , cdept VARCHAR(4) NOT NULL
                    , PRIMARY KEY (cno)
                    ) ;

?section insert
INSERT INTO course VALUES ('C11', 'Intro to CS','for Rookies',3, 100, 'CIS') ;
INSERT INTO course VALUES ('C22', 'Data Structures','Very Useful',3, 50, 'CIS') ;
INSERT INTO course VALUES ('C33', 'Discrete Mathematics', 'Absolutely Necessary',3,
0,'CIS') ;

?section select
SELECT * FROM course ;

?section delete
PURGEDATA course;
```

To run only the commands in section `create`, execute the following:

```
SQL> OBEY C:\Command Interfaces\course.sql (create)

SQL> ?section create
SQL> CREATE TABLE course
+>(
+> cno VARCHAR(3) NOT NULL,
+> cname VARCHAR(22) NOT NULL,
+> cdescp VARCHAR(25) NOT NULL,
+> cred INT,
+> clabfee NUMERIC(5,2),
+> cdept VARCHAR(4) NOT NULL,
+> PRIMARY KEY (cno)
+>) ;

--- SQL Operation complete.
```

To run only the commands in the `insert` section, execute the following:

```
SQL> OBEY C:\Command Interfaces\course.sql (insert)

SQL> ?section insert
SQL> INSERT INTO course VALUES
+> ('C11', 'Intro to CS','For Rookies',3, 100, 'CIS');

--- 1 row(s) inserted.

SQL> INSERT INTO course VALUES
+> ('C22', 'Data Structures','Very Useful',3, 50, 'CIS');

--- 1 row(s) inserted.

SQL> INSERT INTO course VALUES
+> ('C33', 'Discrete Mathematics', 'Absolutely Necessary',3, 0, 'CIS');

--- 1 row(s) inserted.
```

- This command executes all files with `.sql` extension:

```
SQL> OBEY c:\trafci\.sql;
SQL> OBEY c:\trafci
```

- This command executes all files beginning with the word `"script"` and contains one character after the word script and ends with `.sql` extension. For example: `script1.sql`, `script2.sql`, `scriptZ.sql`and so on.

```
SQL> OBEY C:\trafci\script?.sql
```

- This command executes all files that contain the word `"test"`. This includes the files that do not end with `.sql` extension.

```
SQL> OBEY C:\trafci\test
```

- This command executes all files that begin with the word `"script"` and contains one character after the word `"script"` and ends with an extension prefixed by a dot. For example: `script1.sql`, `script2.bat`, `scriptZ.txt`, and so on.

```
SQL> OBEY C:\trafci\script?.
```

- This command executes all files that have `.txt` extension in the current directory, the directory in which the command interface was launched.

```
SQL> OBEY .txt;
```

- This command prompts the user to enter the script filename or a pattern. The default value is `*.sql`.

```
SQL> OBEY;

Enter the script filename [.sql]:
```

8.21. PRUN Command

The PRUN command runs script files in parallel.

8.21.1. Syntax

```
PRUN { -d | -defaults }

PRUN
[ { -sd | -scriptsdir } scriptsdirectory ]
[ { -e  | -extension } filedirectory ]
[ { -ld | -logsdir } log-directory ]
[ { -o  | -overwrite } {Y | N}
[ { -c  | -connections } num ]
```

- `-d | -defaults`

  Specify this option to have PRUN use these default settings:

Parameter	Default Setting
`-sd | -scriptsdir`	PRUN searches for the script files in the same directory as the `trafci.sh` or `trafci.cmd` file (*`trafci-installation-directory`*`/trafci/bin` or *`trafci-installation-directory`*`\trafci\bin`).
`-e | -extension`	The file extension is `.sql`.
`-ld | -logsdir`	PRUN places the log files in the same directory as the script files.
`-o | -overwrite`	No overwriting occurs. PRUN keeps the original information in the log files and appends new information at the end of each file.
`-c | -connections`	PRUN uses two connections.

- `{-sd | -scriptsdir}` *`scripts-directory`*

  In this directory, PRUN processes every file with the specified file extension. If you do not specify a directory or if you specify an invalid directory, an error message occurs, and you are prompted to reenter the directory. Before running PRUN, verify that this directory contains valid script files.

- `{-e | -extension}` *`file-extension`*

  Specify the file extension of the script files. The default is `.sql`.

- `{-ld | -logsdir}` *`log-directory`*

  In this directory, PRUN creates a log file for each script file by appending the `.log` extension to the name of the script file. If you do not specify a log file directory, PRUN places the log files in the same directory as the script files.

- `{-o | -overwrite} {y | n}`

  If you specify `y`, PRUN overwrites the contents of existing log files. By default, PRUN keeps the original information in the log files and appends new information at the end of each file.

- `{-c | -connections}` *`num`*

  Enter a number for the maximum number of connections If you do not specify the maximum number of connections, PRUN uses two connections.

8.21.2. Considerations

- You must enter the command on one line. The command does not require an SQL terminator.
- If you execute the PRUN command without any arguments, then TrafCI prompts you for the PRUN arguments. If you specify one or more options, then the PRUN command runs without prompting you for more input. In the non-interactive mode, if any options are not specified, PRUN uses the default values.
- The `-d` or `-defaults` option cannot be specified with any other option.
- The PRUN log files also contain the log end time.
- PRUN does not support the SPOOL or LOG commands. Those commands are ignored in PRUN script files.
- The environment values from the main session (which are available through the SET commands) are propagated to new sessions started via PRUN. However, prepared statements and parameters are bound only to the main user session.
- For a summary of all errors and warnings that occurred during the PRUN operation, go to the error subdirectory in the same directory as the log files (for example, `C:\log\error`) and open the `prun.err.log` summary file.
- For details about the errors that occurred during the execution of a script file, open each individual log file (*`script-file`*`.sql.log`).

8.21.3. Examples

- To use PRUN, enter the PRUN command in the TrafCI session:

```
SQL> PRUN
```

```
Enter  as input to stop the current prun session
-------------------------------------------------
Enter the scripts directory              : c:\ddl_scripts
Enter the script file extension[sql]     :
Enter the logs directory[scripts dir]    : c:\log
Overwrite the log files (y/n)[n]?        : y
Enter the number of connections(2-248)[2]: 3
```

After you enter the number of connections, PRUN starts to process the script files and displays this status:

```
Status: In Progress.......
```

After executing all the script files, PRUN returns a summary of the operation:

```
          ____________________________________________________
                   PARALLELRUN(PRUN) SUMMARY
          ____________________________________________________
          Total files present.............................. 3
          Total files processed............................ 3
          Total queries processed.......................... 40
          Total errors..................................... 4
          Total warnings................................... 0
          Total successes.................................. 36
          Total connections................................ 5
          Total connection failures........................ 0

          Please verify the error log file c:\log\error\prun.err.log
SQL>
```

In the PRUN summary, the `Total queries processed` is the total number of commands that PRUN processes. Those commands can include SQL statements and commands. The total `errors`, `warnings`, and `successes` also include commands other than SQL statements.

- This PRUN command initiates a parallel run operation with the -d option:

```
SQL> PRUN -d
SQL> PRUN -scriptsdir ./prun/sql -e sql -ld ./prun/logs -o y -connections 5

PRUN options are -scriptsdir    c:/_trafci/prun
                 -logsdir       c:/_trafci/prun/logs
                 -extension     sql
                 -overwrite     y
                 -connections   5
Status: Complete


          ____________________________________________________
                   PARALLELRUN(PRUN) SUMMARY
          ____________________________________________________
          Total files present........................... 99
          Total files processed......................... 99
          Total queries processed...................... 198
          Total errors.................................. 0
          Total warnings................................ 0
          Total warnings................................ 0
          Total connections............................. 5
          Total connection failures..................... 0

================================================================================
PRUN completed at May 20, 2105 9:33:21 AM
================================================================================
```

- PRUN can be started in non-interactive mode using the -q parameter of trafci.cmd or trafci.sh, thus requiring no input:

```
trafci.cmd -h 16.123.456.78
-u user1 -p host1
-q "PRUN -sd c:/_trafci/prun -o y -c 3"
```

- PRUN can be started in non-interactive mode from an OBEY file:

```
SQL> OBEY startPrun.txt
SQL> PRUN -sd c:/_trafci/prun -ld c:/_trafci/prun/logs -e sql -o y -c 5

PRUN options are -scriptsdir    c:/_trafci/prun
                 -logsdir       c:/_trafci/prun/logs
                 -extension     sql
                 -overwrite     yes
                 -connections   5
Status: Complete
```

8.22. QUIT Command

The `QUIT` command disconnects from and exits TrafCI.

8.22.1. Syntax

```
QUIT [WITH] [status] [IF {condition}]
```

- *`status`*

  is any 1-byte integer. *`status`* is a shell return value, and the range of allowable values is platform dependent.

- *`condition`*

  is the same as the condition parameter defined for the IF…THEN Command. See Condition Parameters.

8.22.2. Considerations

You must enter the command on one line. The command does not require an SQL terminator.

8.22.3. Examples

- This command disconnects from and exits TrafCI, which disappears from the screen:

  ```
  SQL> QUIT
  ```

- In a script file, the conditional exit command causes the script file to quit running and disconnect from and exit TrafCI when the previously run command returns error code `4082`:

  ```
  SQL> LOG c:\errorCode.log
  SQL> SELECT * FROM employee;
  SQL> QUIT IF errorcode=4082
  SQL> LOG OFF
  ```

These results are logged when error code 4082 occurs:

```
SQL> SELECT * FROM employee;

**** ERROR[4082] Table, view or stored procedure TRAFODION.USR.EMPLOYEE does not
exist or is inaccessible.

SQL> QUIT IF errorcode=4082
```

8.23. RECONNECT Command

The `RECONNECT` command creates a new connection to the Trafodion database using the login credentials of the last successful connection.

8.23.1. Syntax

```
RECONNECT
```

8.23.2. Considerations

The host name (or IP address) and port number, plus the credentials (user name and password), are used from information previously entered. This is the information specified at launch or when the last `CONNECT` command was executed.

If TrafCI was invoked with the `-noconnect` launch parameter, TrafCI prompts you for the values.

8.23.3. Examples

- This command creates a new connection to the Trafodion database using the login credentials of the last successful connection:

```
SQL> RECONNECT

Connected to Trafodion
```

8.24. REPEAT Command

The REPEAT command re-executes a previous command.

8.24.1. Syntax

```
REPEAT [text | [-]number ]
```

- *text*

  specifies the text of the most recently executed command. The command must have been executed beginning with *text*, but *text* need be only as many characters as necessary to identify the command. TrafCI ignores leading blanks.

- *number*

  is an integer that identifies a command in the history buffer. If number is negative, it indicates the position of the command in the history buffer relative to the current command; if number is positive, it is the ordinal number of a command in the history buffer.

The HISTORY command displays the commands or statements in the history buffer. See the HISTORY Command.

8.25. Considerations

- You must enter the command on one line. The command does not require an SQL terminator.
- To re-execute the immediately preceding command, enter REPEAT without specifying a number. If you enter more than one command on a line, then the REPEAT command re-executes only the last command on the line.
- When a command is selected for repeat, and the SQL terminator value has changed since the execution of that command, then TrafCI replaces the SQL terminator in the command with the current SQL terminator value and executes the command.

8.25.1. Examples

- Display the previously executed commands and re-execute the second to the last command:

```
SQL> HISTORY

1> SET IDLETIMEOUT 0
2> LOG ON
3> SET SCHEMA persnl;
4> SELECT * FROM employee;
5> SHOW TABLES
6> SELECT * FROM dept;
7> SHOW VIEWS
8> SELECT *  FROM emplist;

SQL>
SQL> REPEAT -2

SHOW VIEWS
VIEW NAMES
------------------------------------------------------------
EMPLIST  MGRLIST

SQL>
```

- Re-execute the fifth command in the history buffer:

```
SQL> REPEAT 5

SHOW TABLES
TABLE NAMES
-------------------------------------------------------------
DEPT     EMPLOYEE   JOB         PROJECT

SQL>
```

- Re-execute the `SHOW TABLES` command:

```
SQL> REPEAT SHOW

SHOW TABLES
TABLE NAMES
-------------------------------------------------------------
DEPT     EMPLOYEE   JOB         PROJECT

SQL>
```

8.26. RESET LASTERROR Command

The `RESET LASTERROR` command resets the last error code to 0.

8.26.1. Syntax

```
RESET LASTERROR
```

8.26.2. Considerations

You must enter the command on one line. The command does not require an SQL terminator.

8.26.3. Examples

- This command resets the last error in the current session:

```
SQL> SELECT * FROM emp;

**** ERROR[4082]Object TRAFODION.SCH.EMP does not exist or is inaccessible.

SQL> SHOW LASTERROR

LASTERROR 4082

SQL> RESET LASTERROR
SQL> SHOW LASTERROR

LASTERROR 0
```

8.27. RESET PARAM Command

The RESET PARAM command clears all parameter values or a specified parameter value in the current session.

8.27.1. Syntax

```
RESET PARAM [param-name]
```

- *param-name*

  is the name of the parameter for which you specified a value. Parameter names are case-sensitive. For example, the parameter ?pn is not equivalent to the parameter ?PN. *param-name* can be preceded by a question mark (?), such as ?*param-name*.

  If you do not specify a parameter name, all of the parameter values in the current session are cleared.

8.27.2. Considerations

- You must enter the command on one line. The command does not require an SQL terminator.
- To clear several parameter values but not all, you must use a separate RESET PARAM command for each parameter.

8.27.3. Example

- This command clears the setting of the ?sal (salary) parameter, and the SET PARAM command resets it to a new value:

```
SQL> RESET PARAM ?sal +
SQL> SET PARAM ?sal 80000.00
```

For more information, see Reset the Parameters.

8.28. RUN Command

The RUN command executes the previously executed SQL statement. This command does not repeat an interface command.

8.28.1. Syntax

```
RUN
```

8.28.2. Considerations

- You must enter the command on one line.
- The command does not require an SQL terminator.

8.28.3. Example

- This command executes the previously executed SELECT statement:

```
SQL> SELECT COUNT(*) FROM persnl.employee;

(EXPR)
--------------------
62

--- 1 row(s) selected.

SQL> RUN

(EXPR)
--------------------
62

--- 1 row(s) selected.

SQL>
```

8.29. SAVEHIST Command

The SAVEHIST command saves the session history in a user-specified file. The session history consists of a list of the commands that were executed in the TrafCI session before the SAVEHIST command.

8.29.1. Syntax

```
SAVEHIST file-name [CLEAR]
```

- *file-name*

  is the name of a file into which TrafCI stores the session history. If you want the history file to exist outside the local directory where you launch TrafCI (by default, the bin directory), specify the full directory path of the history file. The specified directory must exist before you execute the SAVEHIST command.

- CLEAR

  instructs TrafCI to clear the contents of the specified file before adding the session history to the file.

8.29.2. Considerations

- You must enter the command on one line. The command does not require an SQL terminator.
- If the specified file already exists, TrafCI appends newer session-history information to the file.

8.29.3. Examples

- This command clears the contents of an existing file named history.txt in the local directory (the same directory where you are running TrafCI) and saves the session history in the file:

  ```
  SQL> SAVEHIST history.txt CLEAR
  SQL>
  ```

- This command saves the session history in a file named hist.txt in the specified directory on a Windows workstation:

  ```
  SQL> SAVEHIST c:\log_files\hist.txt
  SQL>
  ```

- This command saves the session history in a file named `hist.txt` in the specified directory on a Linux or UNIX workstation:

```
SQL> SAVEHIST ./log_files/hist.txt
SQL>
```

For more information, see Display Executed Commands.

8.30. SET COLSEP Command

The SET COLSEP command sets the column separator and allows you to control the formatting of the result displayed for SQL queries. The SET COLSEP command specifies a delimiter value to use for separating columns in each row of the results. The default delimiter is " "(white space).

8.30.1. Syntax

```
SET COLSEP [separator]
```

8.30.2. Considerations

- You must enter the command on one line.
- The SET COLSEP command has no effect if the markup is set to HTML,XML, or CSV.

8.30.3. Examples

- This command specifies the separator as a “|”(pipe):

```
SQL> SET COLSEP |
SQL> SHOW COLSEP
COLSEP "|"
SQL> SELECT * FROM employee;

EMPNUM|EMPNAME       |REGNUM|BRANCHNUM|JOB
------|--------------|------|---------|--------
|    1|ROGER GREEN   |    99|        1|MANAGER
|   23|JERRY HOWARD  |     2|        1|MANAGER
|   29|JACK RAYMOND  |     1|        1|MANAGER
|   32|THOMAS RUDLOFF|     5|        3|MANAGER
|   39|KLAUS SAFFERT |     5|        2|MANAGER

--- 5 row(s) selected.
```

8.31. SET FETCHSIZE Command

The `SET FETCHSIZE` command allows you to change the default fetchsize used by JDBC. Setting the value to `0` sets the fetchsize to the default value used in JDBC.

8.31.1. Syntax

```
SET FETCHSIZE _value_
```

- *`value`*

  is an integer representing the fetch size as a number of rows. Zero (`0`) represents the default value of fetch size set in JDBC.

8.31.2. Considerations

- You must enter the command on one line.
- The command does not require an SQL terminator.

8.31.3. Examples

- This command sets the fetchsize to `1`:

```
SQL> SET FETCHSIZE 1
SQL> SHOW FETCHSIZE

FETCHSIZE 1

SQL> SELECT * FROM stream(t1);

C1      C2      C3
------- ------- -------
TEST1   TEST2   TEST3
AAA     BBB     CCC
```

8.32. SET HISTOPT Command

The `SET HISTOPT` command sets the history option and controls how commands are added to the history buffer. By default, commands within a script file are not added to history. If the history option is set to `ALL`, then all the commands in the script file are added to the history buffer. If no options are specified, `DEFAULT` is used.

8.32.1. Syntax

```
SET HISTOPT [ ALL | DEFAULT ]
```

8.32.2. Considerations

You must enter the command on one line.

8.32.3. Examples

- This command shows only the obey commands added to the history buffer.

```
SQL> SHOW HISTOPT

HISTOPT DEFAULT [No expansion of script files]

SQL> OBEY e:\scripts\nobey\insert2.sql

SQL> ?SECTION insert

SQL> SET SCHEMA trafodion.sch;

--- SQL operation complete.

SQL>  INSERT INTO course1 VALUES
+> ('C11', 'Intro to CS','For Rookies',3, 100,'CIS');

--- 1 row(s) inserted.

SQL>  INSERT INTO course1 VALUES
+> ('C55', 'Computer Arch.','VON Neumann''S Mach.',3, 100, 'CIS');

--- 1 row(s) inserted.
```

```
SQL> HISTORY;

1> SHOW HISTOPT
2> OBEY e:\scripts\nobey\insert2.sql
```

- This command shows all the commands added to the history buffer.

```
SQL> SET HISTOPT ALL
SQL> OBEY e:\scripts\nobey\insert2.sql

?SECTION insert

SQL>  set schema trafodion.sch;

--- SQL operation complete.

SQL>  INSERT INTO course1 VALUES
+> ('C11','Intro to CS','For Rookies',3, 100, 'CIS');

---1 row(s) inserted.

SQL>  INSERT INTO course1 VALUES
+> ('C55','Computer Arch.','Von Neumann''s Mach.',3,100, 'CIS');

---1 row(s) inserted.

SQL> HISTORY;

1> SHOW HISTOPT
2> OBEY e:\scripts\nobey\insert2.sql
3> HISTORY;
4> SET HISTOPT ALL
5> SET SCHEMA trafodion.sch;
6> INSERT INTO course1 VALUES
     ('C11','Intro to CS','For Rookies',3, 100, 'CIS');
7> INSERT INTO course1 VALUES
     ('C55','Computer Arch.','Von Neumann''s MACH.',3,100, 'CIS');
```

8.33. SET IDLETIMEOUT Command

The `SET IDLETIMEOUT` command sets the idle timeout value for the current session. The idle timeout value of a session determines when the session expires after a period of inactivity. The default is `30 minutes`.

8.33.1. Syntax

```
SET IDLETIMEOUT value
```

- *`value`*

  is an integer representing the idle timeout value in minutes. Zero represents an infinite amount of time, meaning that the session never expires.

8.33.2. Considerations

- You must enter the command on one line. The command does not require an SQL terminator.
- If you execute this command in a script file, it affects the session in which the script file runs. You can specify this command in `PRUN` script files. However, running this command from a `PRUN` script file does not affect the idle timeout value for the current session.
- To reset the default timeout value, enter this command:

  ```
  SET IDLETIMEOUT 30
  ```

8.33.3. Examples

- This command sets the idle timeout value to four hours:

```
SQL> SET IDLETIMEOUT 240
```

- This command sets the idle timeout value to an infinite amount of time so that the session never expires:

```
SQL> SET IDLETIMEOUT 0
```

- To reset the idle timeout to the default, enter this command:

```
SQL> SET IDLETIMEOUT 30
SQL>
```

For more information, see Set and Show Session Idle Timeout Value.

8.34. SET LIST_COUNT Command

The SET LIST_COUNT command sets the maximum number of rows to be returned by SELECT statements that are executed after this command. The default is zero, which means that all rows are returned.

8.34.1. Syntax

```
SET LIST_COUNT num-rows
```

- *num-rows*

  is a positive integer that specifies the maximum number of rows of data to be displayed by SELECT statements that are executed after this command. Zero means that all rows of data are returned.

8.34.2. Considerations

- You must enter the command on one line. The command does not require an SQL terminator.
- To reset the number of displayed rows, enter this command:

```
SET LIST_COUNT 0
```

8.34.3. Examples

- This command specifies that the number of rows to be displayed by SELECT statements is five:

```
SQL> SET LIST_count 5
SQL> SELECT empnum, first_name, last_name FROM persnl.employee ORDER BY empnum;

EMPNUM FIRST_NAME      LAST_NAME
------ --------------- --------------------
     1 ROGER           GREEN
    23 JERRY           HOWARD
    29 JANE            RAYMOND
    32 THOMAS          RUDLOFF
    39 KLAUS           SAFFERT

--- 5 row(s) selected. LIST_COUNT was reached.

SQL>
```

- This command resets the number of displayed rows to all rows:

```
SQL> SET LIST_COUNT 0
SQL> SELECT empnum, first_name, last_name
+> FROM persnl.employee
+> ORDER BY empnum;

EMPNUM FIRST_NAME      LAST_NAME
------ --------------- --------------------
     1 ROGER           GREEN
    23 JERRY           HOWARD
    29 JANE            RAYMOND
    32 THOMAS          RUDLOFF
    39 KLAUS           SAFFERT
    43 PAUL            WINTER
    65 RACHEL          MCKAY
...
   995 Walt            Farley

--- 62 row(s) selected.

SQL>
```

8.35. SET MARKUP Command

The SET MARKUP command sets the markup format and controls how results are displayed by TrafCI.

8.35.1. Syntax

```
SET MARKUP [ RAW | HTML | XML | CSV | COLSEP ]
```

The supported options enable results to be displayed in XML, HTML, CSV (Comma Separated Values), and COLSEP format. The default format is RAW.

8.35.2. Considerations

- You must enter the command on one line.
- If the MARKUP format is CSV or COLSEP, the column header information and status messages are not displayed.
- For the XML and HTML markup format, the syntax and interface errors is consistent XML and HTML markup is displayed.
- For XML markup, any occurrence of]]> that appear in the error message or invalid query are replaced with]]>.
- When error messages are output as HTML markup, both the > (greater than) and < (less than) symbols are replaced with their escaped versions: > and <, respectively. An example of the formatted error messages are show below.

8.35.3. Examples

- This command specifies results be displayed in `HTML`:

```
SQL> SET MARKUP HTML
SQL> SELECT c.custnum, c.custnum, ordernum, order_date
+> FROM customer c, orders o where c.custnum=o.custnum;

<TABLE>
<!--SELECT c.custnum, c.custname,ordernum,order_date
FROM customer c, orders o where c.custnum=o.custnum;-->
<tr>
  <th>CUSTNUM</th>
  <th>CUSTNAME</th>
  <th>ORDERNUM</th>
  <th>ORDER_DATE</th>
</tr>
<tr>
  <td>143</td>
  <td>STEVENS SUPPLY</td>
  <td>700510</td>
  <td>2105-05-01</td>
</tr>
<tr>
  <td>3333</td>
  <td>NATIONAL UTILITIES</td>
  <td>600480</td>
  <td>2105-05-12</td>
</tr>
<tr>
  <td>7777</td>
  <td>SLEEP WELL HOTELS</td>
  <td>100250</td>
  <td>2105-01-23</td>
</tr>
<!-- --- 3 row(s) selected.-->
</TABLE>
```

```
SQL> SELECT c.custnum, c.custname,ordernum,order_date,
+> FROM customer c, orders o where c.custnum=o.custnum;

<TABLE>
<!-- SELECT c.custnum, c.custname,ordernum,order_date,
FROM customer c, orders o where c.custnum=o.custnum;-->
<tr>
  <th>Error Id</th>
  <th>Error Code</th>
  <th>Error Message</th>
</tr>
<tr>
  <td>1</td>
  <td>4082</td>
  <td>Object TRAFODION.NVS.CUSTOMER does not exist or is inaccessible.</td>
</tr>
</TABLE>
```

- To set the application to format output as `HTML`:

```
SQL> SET MARKUP HTML
```

HTML formatted error message example:

```
SQL> SET MARKUP <invalid>

<?xml version="1.0"?>
<Results>
  <Query>
    <![CDATA[set markup <invalid ]]>
  </Query>
  <ErrorList>
    <Error id="1">
    <ErrorCode>NVCI001</ErrorCode>
    <ErrorMsg> <![CDATA[
ERROR: A syntax error occurred at or before:
set markup <invalid>
           ^ ]]
    </ErrorMsg>
  </ErrorList>
</Results>
```

- This command specifies results be displayed in CSV:

```
SQL> SET MARKUP CSV
SQL> SELECT c.custnum, c.custnum, ordernum, order_date
+> FROM customer c,orders o where c.custnum=o.custnum;

143,STEVENS SUPPLY ,700510,2105-05-01
3333,NATIONAL UTILITIES,600480,2105-05-12
7777,SLEEPWELL HOTELS ,100250,2105-01-23
324,PREMIER INSURANCE ,500450,2105-04-20
926,METALL-AG. ,200300,2105-02-06
123,BROWN MEDICAL CO ,200490,2105-03-19
123,BROWN MEDICAL CO ,300380,2105-03-19
543,FRESNO STATE BANK ,300350,2105-03-03
5635,ROYAL CHEMICALS ,101220,2105-05-21
21,CENTRAL UNIVERSITY,200320,2105-02-17
1234,DATASPEED ,100210,2105-04-10
3210,BESTFOOD MARKETS ,800660,2105-05-09
```

- This command specifies results be displayed in XML:

```
SQL> SET MARKUP XML
SQL> SELECT * FROM author

<?xml version="1.0"?>
<Results>
  <Query>
    <![CDATA[select  from author;]]>
  </Query>
  <rowid="1">
    <AUTHORID>91111</AUTHORID>
    <AUTHORNAME>Bjarne Stroustrup</AUTHORNAME>
  </row>
  <rowid="2">
    <AUTHORID>444444</AUTHORID>
    <AUTHORNAME>John Steinbeck</AUTHORNAME>
  </row>
  <rowid="3">
    <AUTHORID>2323423</AUTHORID>
    <AUTHORNAME>Irwin Shaw</AUTHORNAME>
  </row>
  <rowid="4">
    <AUTHORID>93333</AUTHORID>
    <AUTHORNAME>Martin Fowler</AUTHORNAME>
  </row>
  <rowid="5">
    <AUTHORID>92222</AUTHORID>
    <AUTHORNAME>Grady Booch</AUTHORNAME>
  </row>
  <rowid="6">
    <AUTHORID>84758345</AUTHORID>
    <AUTHORNAME>Judy Blume</AUTHORNAME>
  </row>
  <rowid="7">
    <AUTHORID>89832473</AUTHORID>
    <AUTHORNAME>Barbara Kingsolver</AUTHORNAME>
  </row>
  <Status> <![CDATA[-- 7 row(s) selected .]]></Status>
</Results>
```

- To set the application to format output as XML:

```
SQL> SET MARKUP XML
```

XML formatted error message examples:

```
SQL> SET MARKUP <]]>

<?xml version="1.0"?>
<Results>
  <Query>
    <![CDATA[set markup <]]&#62; ]]>>
  </Query>
  <ErrorList>
    <Error id="1">
    <ErrorCode>UNKNOWN ERROR CODE</ErrorCode
    <ErrorMessage> <![CDATA[
ERROR: A syntax error occurred at or before:
set markup <]]&#62;>
           ^ ]]<>
    </ErrorMessage>
  </ErrorList>
</Results>
```

- This command displays CSV like output using the COLSEP value as a separator.

```
SQL> SET COLSEP |
SQL> SET MARKUP COLSEP
SQL> SELECT * FROM employee;

32|THOMAS       |RUDLOFF      |2000|100|138000.40
39|KLAUS        |SAFFERT      |3200|100|75000.00
89|PETER        |SMITH        |3300|300|37000.40
29|JANE         |RAYMOND      |3000|100|136000.00
65|RACHEL       |MCKAY        |4000|100|118000.00
75|TIM          |WALKER       |3000|300|320000.00
11|ROGER        |GREEN        |9000|100|175500.00
93|DONALD       |TAYLOR       |3100|300|33000.00
```

8.36. SET PARAM Command

The SET PARAM command associates a parameter name with a parameter value in the current session. The parameter name and value are associated with one of these parameter types:

- Named parameter (represented by ?*param-name*) in a DML statement or in a prepared SQL statement
- Unnamed parameter (represented by ?) in a prepared SQL statement only

A prepared statement is one that you SQL compile by using the PREPARE statement. For more information about PREPARE, see the *Trafodion SQL Reference Manual*.

After running SET PARAM commands in the session:

- You can specify named parameters (?*param-name*) in a DML statement.
- You can execute a prepared statement with named parameters by using the EXECUTE statement without a USING clause.
- You can execute a prepared statement with unnamed parameters by using the EXECUTE statement with a USING clause that contains literal values and/or a list of the named parameters set by SET PARAM.

The EXECUTE statement substitutes parameter values for the parameters in the prepared statement. For more information about EXECUTE, see the *Trafodion SQL Reference Manual*.

8.36.1. Syntax

```
SET PARAM param-name [UTF8] param-value
```

- *param-name*

  is the name of the parameter for which a value is specified. Parameter names are case-sensitive. For example, the parameter ?pn is not equivalent to the parameter ?PN. *param-name* can be preceded by a question mark (?), such as ?*param-name*.

- UTF8

  specifies that a character string specified for the parameter value, *param-value*, uses the UTF8 character set. If the character string is in UTF8 format, it must be prefixed by UTF8.

- *param-value*

  is a numeric or character literal that specifies the value for the parameter. If you do not specify a value, TrafCI returns an error.

  If *param-value* is a character literal and the target column type is a character string, you do not have to enclose the value in single quotation marks. Its data type is determined from the data type of the column to which the literal is assigned. Character strings specified as parameter values are always case-sensitive even if they are not enclosed in quotation marks. If the character string is in UTF8 format, it must be prefixed by UTF8.

8.36.2. Considerations

- You must enter the command on one line. The command does not require an SQL terminator.
- Use separate SET PARAM commands to name and assign values to each unique parameter in a prepared SQL statement before running the EXECUTE statement.
- Parameter names are case-sensitive. If you specify a parameter name in lowercase in the SET PARAM command, you must specify it in lowercase in other statements, such as DML statements or EXECUTE.
- The name of a named parameter (?*param-name*) in a DML statement must be identical to the parameter name (*param-name*) that you specify in a SET PARAM command.

8.36.3. Examples

- This command sets a value for the `?sal` (`salary`) parameter:

```
SQL> SET PARAM ?sal 40000.00
```

- This command sets a character string value, `GREEN`, for the `?lastname` parameter:

```
SQL> SET PARAM ?lastname GREEN
```

- These commands set values for named parameters in a subsequent `SELECT` statement:

```
SQL> SET PARAM ?sal 80000.00
SQL> SET PARAM ?job 100
SQL> SELECT * FROM persnl.employee WHERE salary = ?sal AND jobcode = ?job;

EMPNUM FIRST_NAME      LAST_NAME            DEPTNUM JOBCODE SALARY
------ --------------- -------------------- ------- ------- ----------
    72 GLENN           THOMAS                  3300     100   80000.00

--- 1 row(s) selected.

SQL>
```

The names of the named parameters, `?sal` and `?job`, in the `SELECT` statement are identical to the parameter `names`, `sal` and `job`, in the `SET PARAM` command.

- This command sets a character string value, `Peña`, which is in `UTF8` format, for the `?lastname` parameter:

```
SQL> SET PARAM ?lastname UTF8'Pe&#241;a'
```

- This command sets a character string value, which uses the `UTF8` character set and is in hexadecimal notation, for the `?lastname` parameter:

```
SQL> SET PARAM ?lastname UTF8x'5065266e74696c64653b61'
```

For more information, see Set Parameters.

8.37. SET PROMPT Command

The `SET PROMPT` command sets the prompt of the current session to a specified string and/or to the session variables, which start with `%`. The default prompt is `SQL>`.

8.37.1. Syntax

```
SET PROMPT [string] [%USER] [%SERVER] [%SCHEMA]
```

- *`string`*

  is a string value to be displayed as the prompt. The string may contain any characters. Spaces are allowed if you enclose the string in double quotes (`"`). If you do not enclose the string in double quotes, the prompt is displayed in uppercase.

- `%USER`

  displays the session user name as the prompt.

- `%SERVER`

  displays the session host name and port number as the prompt.

- `%SCHEMA`

  displays the session schema as the prompt.

8.37.2. Considerations

- You must enter the command on one line. The command does not require an SQL terminator.
- To reset the default prompt, enter this command:

  ```
  SET PROMPT
  ```

8.37.3. Examples

- This `SET PROMPT` command sets the SQL prompt to `ENTER>`:

```
SQL> SET PROMPT Enter>
ENTER>
```

- To reset the SQL prompt to the default, enter this `SET PROMPT` command:

```
ENTER> SET PROMPT +
SQL>
```

- This command displays the session user name for the prompt:

```
SQL> SET PROMPT %user>
user1>
```

- This command displays the session host name and port number for the prompt:

```
SQL> SET PROMPT %server>
sqws135.houston.host.com:22900>
```

- This command displays the session schema for the prompt:

```
SQL> SET PROMPT "Schema %schema:"
Schema USR:
```

- This command displays multiple session variables:

```
SQL> SET PROMPT %USER@%SCHEMA> user1@USR>
user1@USR>set prompt %SERVER:%USER>
sqws135.houston.host.com:22900:user1>
sqws135.houston.host.com:22900:user1> SET PROMPT "%schema CI> "
USR CI>
```

For more information, see Customize Standard Prompt.

8.38. SET SQLPROMPT Command

The `SET SQLPROMPT` command sets the SQL prompt of the current session to a specified string. The default is `SQL>`.

8.38.1. Syntax

```
SET SQLPROMPT [string] [%USER] [%SERVER] [%SCHEMA]
```

- *`string`*

  is a string value to be displayed as the SQL prompt. The string may contain any characters. Spaces are allowed if you enclose the string in double quotes. If you do not enclose the string in double quotes ("), the prompt is displayed in uppercase.

- `%USER`

  displays the session user name as the prompt.

- `%SERVER`

  displays the session host name and port number as the prompt.

- `%SCHEMA`

  displays the session schema as the prompt.

8.38.2. Considerations

- You must enter the command on one line. The command does not require an SQL terminator.
- To reset the default SQL prompt, enter this command:

  ```
  SET SQLPROMPT
  ```

8.38.3. Examples

- This command sets the SQL prompt to `ENTER>`:

```
SQL> SET SQLPROMPT Enter>
ENTER>
```

- To reset the SQL prompt to the default, enter this command:

```
ENTER> SET SQLPROMPT
SQL>
```

- This command displays the session user name for the prompt:

```
SQL> SET SQLPROMPT %user>
user1>
```

- This command displays the session host name and port number for the prompt:

```
SQL> SET SQLPROMPT %server>
sqws135.houston.host.com:22900>
```

- This command displays the session schema for the prompt:

```
SQL> SET SQLPROMPT "Schema %schema:"
Schema USR:
```

- This command displays multiple session variables:

```
SQL> SET SQLPROMPT %USER@%SCHEMA>
user1@USR>

SQL> SET SQLPROMPT %SERVER:%USER>
sqws135.houston.host.com:22900:user1>
sqws135.houston.host.com:22900:user1> SET SQLPROMPT "%schema CI> "
USR CI>
```

For more information, see Customize Standard Prompt.

8.39. SET SQLTERMINATOR Command

The `SET SQLTERMINATOR` command sets the SQL statement terminator of the current session. The default is a semicolon (`;`).

8.39.1. Syntax

```
SET SQLTERMINATOR string
```

- *`string`*

  is a string value for the SQL terminator. The string may contain any characters except spaces. Spaces are disallowed even if you enclose the string in double quotes. Lowercase and uppercase characters are accepted, but the SQL terminator is always shown in uppercase.

8.39.2. Considerations

- You must enter the command on one line. The command does not require an SQL terminator.
- Do not include a reserved word as an SQL terminator.
- If you execute this command in a script file, it affects not only the SQL statements in the script file but all subsequent SQL statements that are run in the current session. If you set the SQL terminator in a script file, reset the default terminator at the end of the script file.
- To reset the default SQL terminator (`;`), enter this command:

  ```
  SET SQLTERMINATOR ;
  ```

8.39.3. Examples

- This command sets the SQL terminator to a period (`.`):

```
SQL> SET SQLTERMINATOR .
```

- This command sets the SQL terminator to a word, `go`:

```
SQL> SET SQLTERMINATOR go
```

  This query ends with the new terminator, `go`:

```
SQL> SELECT * FROM persnl.employee go
```

- To reset the SQL terminator to the default, enter this command:

```
SQL> SET SQLTERMINATOR ;
```

For more information, Set and Show the SQL Terminator.

8.40. SET STATISTICS Command

The `SET STATISTICS` command automatically retrieves the statistics information for a query being executed. The results returned are the same as would have been returned if the `GET STATISTICS` command was executed. The default is `OFF` which means the statistics information is not automatically printed for any queries.

8.40.1. Syntax

```
SET STATISTICS { ON | OFF }
```

8.40.2. Considerations

You must enter the command on one line.

8.40.3. Examples

- This command shows the default output format as PERTABLE:

```
SQL> SET STATISTICS ON
SQL> SELECT * FROM job;

JOBCODE JOBDESC
------- ------------------
100     MANAGER
450     PROGRAMMER
900     SECRETARY
300     SALESREP
500     ACCOUNTANT
400     SYSTEM ANALYST
250     ASSEMBLER
420     ENGINEER
600     ADMINISTRATOR
200     PRODUCTION SUPV

--- 11 row(s) selected.

Start Time             2105/05/18 21:45:34.082329
End Time               2105/05/18 21:45:34.300265
Elapsed Time                      00:00:00.217936
Compile Time                      00:00:00.002423
Execution Time                    00:00:00.218750

Table Name   Records  Records  Disk  Message  Message  Lock  Lock  Disk Process
            Accessed     Used  I/Os    Count    Bytes  Escl  Wait     Busy Time
TRAFODION.TOI.JOB
                  2        2     0        4    15232     0     0           363


SQL>
```

For more information on the STATISTICS command, see the *Trafodion SQL Reference Manual*.

8.41. SET TIME Command

The `SET TIME` command causes the local time of the client workstation to be displayed as part of the interface prompt. By default, the local time is not displayed in the interface prompt.

8.41.1. Syntax

```
SET TIME { ON[12H] | OFF }
```

- `ON`

  specifies that the local time be displayed as part of the prompt.

- `OFF`

  specifies that the local time not be displayed as part of the prompt. `OFF` is the default.

8.41.2. Considerations

- You must enter the command on one line. The command does not require an SQL terminator.
- The default is a 24-hour military style display. The additional argument of `12h` allows the time to be displayed in a 12-hour AM/PM style.

8.41.3. Examples

- This command causes the local time to be displayed in the SQL prompt:

```
SQL> SET TIME ON
14:17:17 SQL>
```

- This command causes the local time to be displayed in 12-hour AM/PM style in the SQL prompt:

```
SQL> SET TIME ON 12H
2:17:17 PM SQL>
```

- This command turns off the local time in the SQL prompt:

```
2:17:17 PM SQL> SET TIME OFF
SQL>
```

For more information, see Customize the Standard Prompt.

8.42. SET TIMING Command

The `SET TIMING` command causes the elapsed time to be displayed after each SQL statement executes. This command does not cause the elapsed time of interface commands to be displayed. By default, the elapsed time is `off`.

8.42.1. Syntax

```
SET TIMING { ON | OFF }
```

- `ON`

  specifies the elapsed time be displayed after each SQL statement executes.

- `OFF`

  specifies that the elapsed time not be displayed after each SQL statement executes. `OFF` is the default.

8.42.2. Considerations

- You must enter the command on one line. The command does not require an SQL terminator.
- The elapsed time value includes compile and execution time plus any network I/O time and client-side processing time.

8.42.3. Examples

- This command displays the elapsed time of SQL statements:

  ```
  SQL> SET TIMING ON
  ```

- This command turns off the elapsed time:

  ```
  SQL> SET TIMING OFF
  ```

For more information, see Display the Elapsed Time.

8.43. SHOW ACTIVITYCOUNT Command

The `SHOW ACTIVITYCOUNT` command provides an alias for `SHOW RECCOUNT`. `ACTIVITYCOUNT` is an alias for `RECCOUNT`. For more information, see the SHOW RECCOUNT Command.

8.43.1. Syntax

```
SHOW ACTIVITYCOUNT
```

8.43.2. Examples

- This command shows the record count of the previous executed SQL statement:

```
SQL> SHOW ACTIVITYCOUNT

ACTIVITYCOUNT 0
```

8.44. SHOW ALIAS Command

The SHOW ALIAS command displays all or a set of aliases available in the current TrafCI session. If a pattern is specified, then all aliases matching the pattern are displayed. By default, all aliases in the current session are displayed.

8.44.1. Syntax

```
SHOW ALIAS [ alias-name | wild-card-pattern ]
```

- *alias-name*

  is any alias name that is used with the ALIAS command. See ALIAS Command.

- *wild-card-pattern*

  is a character string used to search for and display aliases with names that match the character string. *wild-card-pattern* matches an uppercase string unless you enclose it within double quotes. To look for similar values, specify only part of the characters of *wild-card-pattern* combined with these wild-card characters.

%	Use a percent sign (%) to indicate zero or more characters of any type. For example, %art% matches SMART, ARTIFICIAL, and PARTICULAR but not smart or Hearts. "%art%" matches smart and Hearts but not SMART, ARTIFICIAL, or PARTICULAR.
*	Use an asterisk () to indicate zero or more characters of any type. For example, *art matches SMART, ARTIFICIAL, and PARTICULAR but not smart or Hearts. "**art**" matches smart and Hearts but not SMART, ARTIFICIAL, or PARTICULAR.
_	Use an underscore (_) to indicate any single character. For example, boo_ matches BOOK and BOOT but not BOO or BOOTS. "boo_" matches book and boot but not boo or boots.
?	Use a question mark (?) to indicate any single character. For example, boo? matches BOOK and BOOT but not BOO or BOOTS. "boo?" matches book and boot but not boo or boots.

8.44.2. Considerations

You must enter the command on one line. The command does not require an SQL terminator.

8.44.3. Examples

- This command displays a list of the available aliases:

```
SQL> SHOW ALIAS

.OS AS LH
.GOTO AS GOTO
USE AS SET SCHEMA
```

- This command displays the `.GOTO` alias:

```
SQL> SHOW ALIAS .GOTO

.GOTO AS GOTO
```

- This command displays the `.FOO` alias:

```
SQL> SHOW ALIAS .FOO

No aliases found.
```

- This command displays all aliases beginning with the letter S:

```
SQL> SHOW ALIAS S*

SEL AS SELECT
SHOWTIME AS SHOW TIME
ST AS SHOW TABLES
```

8.45. SHOW ALIASES Command

The SHOW ALIASES command displays all the aliases available in the current TrafCI session.

8.45.1. Syntax

```
SHOW ALIASES
```

8.45.2. Considerations

You must enter the command on one line. The command does not require an SQL terminator.

8.45.3. Examples

- This command displays all the aliases in the current TrafCI session:

```
SQL> SHOW ALIASES

.OS AS LH
.GOTO AS GOTO
USE AS SET SCHEMA
```

8.46. SHOW CATALOG Command

The `SHOW CATALOG` command displays the current catalog of the TrafCI session.

8.46.1. Syntax

```
SHOW CATALOG
```

8.46.2. Considerations

- You must enter the command on one line. The command does not require an SQL terminator.
- If the `SET TIMING` command is set to `ON`, the elapsed time information is displayed.

8.46.3. Example

- This command shows that the current catalog of the session is TRAFODION:

```
SQL> SHOW CATALOG

CATALOG TRAFODION
```

8.47. SHOW COLSEP Command

The SHOW COLSEP command displays the value of the column separator for the current TrafCI session.

8.47.1. Syntax

```
SHOW COLSEP
```

8.47.2. Considerations

- You must enter the command on one line. The command does not require an SQL terminator.
- If the SET TIMING command is set to ON, the elapsed time information is displayed.

8.47.3. Examples

- This command displays the column separator.

```
SQL> SHOW COLSEP

COLSEP " "

SQL> SET COLSEP
SQL> SHOW COLSEP

COLSEP ""
```

- This command displays the column separator.

```
SQL> SHOW COLSEP

COLSEP " "

SQL> SET COLSEP
SQL> SHOW COLSEP

COLSEP ""
```

8.48. SHOW ERRORCODE Command

The `SHOW ERRORCODE` command is an alias for the `SHOW LASTERROR` command. `ERRORCODE` is an alias for `LASTERROR`. For more information, see SHOW LASTERROR Command.

8.48.1. Syntax

```
SHOW ERRORCODE
```

8.48.2. Examples

- This command displays the error of the last SQL statement that was executed:

```
SQL> SHOW ERRORCODE

ERRORCODE 29481
```

8.49. SHOW FETCHSIZE Command

The `SHOW FETCHSIZE` command displays the fetch size value for the current TrafCI session.

8.49.1. Syntax

```
SHOW FETCHSIZE
```

8.49.2. Considerations

You must enter the command on one line.

8.49.3. Examples

- These commands display the fetch size in the current TrafCI session, set the fetch size to a new value, and then redisplay the fetch size:

```
SQL> SHOW FETCHSIZE

FETCHSIZE 0 [Default]

SQL> SET FETCHSIZE 1
SQL> SHOW FETCHSIZE

FETCHSIZE 1
```

8.50. SHOW HISTOPT Command

The `SHOW HISTOPT` command displays the value that has been set for the history option.

8.50.1. Syntax

```
SHOW HISTOPT
```

8.50.2. Considerations

- You must enter the command on one line.
- If the `SET TIMING` command is set to `ON`, the elapsed time information is displayed.

8.50.3. Examples

- This command displays the value set for the history option:

```
SQL> SHOW HISTOPT

HISTOPT DEFAULT [No expansion of script files]

SQL> SET HISTOPT ALL
SQL> SHOW HISTOPT

HISTOPT ALL
```

8.51. SHOW IDLETIMEOUT Command

The `SHOW IDLETIMEOUT` command displays the idle timeout value of the current TrafCI session. The idle timeout value of a session determines when the session expires after a period of inactivity. The default is `30 minutes`.

8.51.1. Syntax

```
SHOW IDLETIMEOUT
```

8.51.2. Considerations

- You must enter the command on one line. The command does not require an SQL terminator.
- If the `SET TIMING` command is set to `ON`, the elapsed time information is displayed.

8.51.3. Examples

- This command shows that the idle timeout value of the session is 30 minutes, which is the default:

```
SQL> SHOW IDLETIMEOUT

IDLETIMEOUT 30 min(s)

Elapsed time:00:00:00:078
```

- This command shows that the idle timeout value of the session is four hours:

```
SQL> SHOW IDLETIMEOUT

IDLETIMEOUT 240 min(s)
```

- This command shows that the idle timeout value is an infinite amount of time, meaning that the session never expires:

```
SQL> SHOW IDLETIMEOUT

IDLETIMEOUT 0 min(s) [Never Expires]
```

- This command displays the elapsed time information because `SET TIMING` command is enabled:

```
SQL> SET TIMING ON
SQL> SHOW IDLETIMEOUT

IDLETIMEOUT 0 min(s) [Never Expires]

Elapsed time:00:00:00:078
```

For more information, see Set and Show Session Idle Timeout Value.

8.52. SHOW LASTERROR Command

The `SHOW LASTERROR` command displays the error of the last SQL statement that was executed. If the query was successful, then `0` is returned; otherwise an SQL error code is returned.

8.52.1. Syntax

```
SHOW LASTERROR
```

8.52.2. Considerations

- You must enter the command on one line. The command does not require an SQL terminator.
- If the `SET TIMING` command is set to `ON`, the elapsed time information is displayed.

8.52.3. Examples

- This command shows the last error in the current session:

```
SQL> SELECT * FROM emp;

**** ERROR[4082]Object TRAFODION.SCH.EMP does not exist or is inaccessible.

SQL> SHOW LASTERROR

LASTERROR 4082
```

8.53. SHOW LIST_COUNT Command

The SHOW LIST_COUNT command displays the maximum number of rows to be returned by SELECT statements in the current TrafCI session. The default is zero, which means that all rows are returned.

8.53.1. Syntax

```
SHOW LIST_COUNT
```

8.53.2. Considerations

- You must enter the command on one line. The command does not require an SQL terminator.
- If the SET TIMING command is set to ON, the elapsed time information is displayed.

8.53.3. Examples

- This command shows that SELECT statements return all rows in the current session:

```
SQL> SHOW LIST_COUNT

LISTCOUNT 0 [All Rows]

Elapsed time:00:00:00:078
```

- This command shows that the maximum number of rows to be displayed by SELECT statements in the session is five:

```
SQL> SET LIST_COUNT 5
SQL> SHOW LIST_COUNT

LIST_COUNT 5

Elapsed time:00:00:00:078
```

8.54. SHOW MARKUP Command

The `SHOW MARKUP` command displays the value set for the markup option.

8.54.1. Syntax

```
SHOW MARKUP
```

8.54.2. Considerations

- You must enter the command on one line.
- If the `SET TIMING` command is set to `ON`, the elapsed time information is displayed.

8.54.3. Examples

- This command displays the value set for the markup option:

```
SQL> SHOW MARKUP

MARKUP RAW

Elapsed time:00:00:00:078
```

8.55. SHOW PARAM Command

The SHOW PARAM command displays the parameters that are set in the current TrafCI session.

8.55.1. Syntax

```
SHOW PARAM
```

8.55.2. Considerations

- You must enter the command on one line. The command does not require an SQL terminator.
- If the SET TIMING command is set to ON, the elapsed time information is displayed.

8.55.3. Example

- This command shows that parameters that are set for the current session:

```
SQL> SHOW PARAM

lastname GREEN
dn 1500
sal 40000.00
```

- This command shows that when no parameters exist, the SHOW PARAM command displays an error message:

```
SQL> SHOW PARAM

No parameters found.
```

For more information, Display Session Parameters.

8.56. SHOW PREPARED Command

The `SHOW PREPARED` command displays the prepared statements in the current TrafCI session. If a pattern is specified, then all prepared statements matching the prepared statement name pattern are displayed. By default, all prepared statements in the current session are displayed.

8.56.1. Syntax

```
SHOW PREPARED
```

8.56.2. Considerations

- You must enter the command on one line. The command does not require an SQL terminator.
- If the `SET TIMING` command is set to `ON`, the elapsed time information is displayed.

8.56.3. Examples

- This command shows all the prepared statements, by default:

```
SQL> SHOW PREPARED

S1
       SELECT *  FROM t1
S2
       SELECT * FROM student
T1
       SELECT * FROM test123

SQL> SHOW PREPARED s%

S1
       SELECT * FROM t1
S2
       SELECT * FROM student

SQL> SHOW PREPARED t%

T1
       SELECT * FROM test123
```

8.57. SHOW RECCOUNT Command

The `SHOW RECCOUNT` command displays the record count of the previously executed SQL statement. If the previously executed command was an interface command, then TrafCI returns zero.

8.57.1. Syntax

```
SHOW RECCOUNT
```

8.57.2. Considerations

- You must enter the command on one line. The command does not need an SQL terminator.
- If the `SET TIMING` command is set to `ON`, the elapsed time information is displayed.

8.57.3. Examples

- This command displays the record count of the SQL statement that was executed last:

```
SQL> SELECT * FROM employee;
SQL> SHOW RECCOUNT

RECCOUNT 62
```

8.58. SHOW REMOTEPROCESS Command

The `SHOW REMOTEPROCESS` command displays the process name of the DCS server that is handling the current connection.

8.58.1. Syntax

```
SHOW REMOTEPROCESS
```

8.58.2. Considerations

- You must enter the command on one line. The command does not need an SQL terminator.
- The command does not need an SQL terminator.

8.58.3. Example

- This command displays the process name, `\g4t3028.houston.host.com:0.$Z0000M2`, of the DCS server that is handling the current connection:

```
SQL> SHOW REMOTEPROCESS

REMOTE PROCESS \g4t3028.houston.host.com:0.$Z0000M2

SQL>
```

8.59. SHOW SCHEMA Command

The SHOW SCHEMA command displays the current schema of the TrafCI session.

8.59.1. Syntax

```
SHOW SCHEMA
```

8.59.2. Considerations

- You must enter the command on one line. The command does not require an SQL terminator.
- If the SET TIMING command is set to ON, the elapsed time information is displayed.

8.59.3. Example

- This command shows that the current schema of the session is PERSNL:

```
SQL> SHOW SCHEMA

SCHEMA PERSNL
```

For more information, see Set and Show the Current Schema.

8.60. SHOW SESSION Command

`SHOW SESSION` or `SESSION` displays attributes of the current TrafCI session. You can also use the `ENV` command to perform the same function.

8.60.1. Syntax

```
[SHOW] SESSION
```

8.60.2. Considerations

- You must enter the command on one line. The command does not require an SQL terminator.
- If the `SET TIMING` command is set to `ON`, the elapsed time information is displayed.
- `SHOW SESSION` or `SESSION` displays these attributes:

Attribute	Description
`COLSEP`	Current column separator, which is used to control how query results are displayed. For more information, SET COLSEP Command.
`HISTOPT`	Current history options, which controls how the commands are added to the history buffer. For more information, see SET HISTOPT Command.
`IDLETIMEOUT`	Current idle timeout value, which determines when the session expire after a period of inactivity. By default, the idle timeout is `30 minutes`. For more information, see Set and Show Session Idle Timeout Value and SET IDLETIMEOUT Command.
`LIST_COUNT`	Current list count, which is the maximum number of rows that can be returned by SELECT statements. By default, the list count is all rows. For more information, see SET LIST_COUNT Command.
`LOG FILE`	Current log file and the directory containing the log file. By default, logging during a session is turned off. For more information, see Log Output, and LOG Command.
`LOG OPTIONS`	Current logging options. By default, logging during a session is turned off, and this attribute does not appear in the output. For more information, see the LOG Command or SPOOL Command.
`MARKUP`	Current markup option selected for the session. The default option is RAW. For more information, see SET MARKUP Command".
`PROMPT`	Current prompt for the session. For example, the default is `SQL>`. For more information, see Customize the Standard Prompt and SET PROMPT Command.

Attribute	Description
SCHEMA	Current schema. The default is USR. For more information, see Set and Show the Current Schema.
SERVER	Host name and port number that you entered when logging in to the database platform. For more information, see Log In to Database Platform.
SQLTERMINATOR	Current SQL statement terminator. The default is a semicolon (;). For more information, see Set and Show the SQL Terminator and SHOW SQLTERMINATOR Command.
STATISTICS	Current setting (on or off) of statistics. For more information, see the SET STATISTICS Command.
TIME	Current setting (on or off) of the local time as part of the prompt. When this command is set to on, military time is displayed. By default, the local time is off. For more information, see Customize the Standard Prompt and SET TIME Command.
TIMING	Current setting (on or off) of the elapsed time. By default, the elapsed time is off. For more information, see Display the Elapsed Time and SET TIMING Command.
USER	User name that you entered when logging in to the database platform. For more information, see Log In to Database Platform.

8.60.3. Examples

- This SHOW SESSION command displays the attributes of the current session:

```
SQL> SHOW SESSION

COLSEP              " "
HISTOPT             DEFAULT [No expansion of script files]
IDLETIMEOUT         0 min(s) [Never Expires]
LIST_COUNT          0 [All Rows]
LOG FILE            c:\session.txt
LOG OPTIONS         APPEND,CMDTEXT ON
MARKUP              RAW
PROMPT              SQL>
SCHEMA              SEABASE
SERVER              sqws135.houston.host.com:37800
SQLTERMINATOR       ;
STATISTICS          OFF
TIME                OFF
TIMING              OFF
USER                user1
```

- This SESSION command shows the effect of setting various session attributes:

```
SQL> SESSION

COLSEP              " "
HISTOPT             DEFAULT [No expansion of script files]
IDLETIMEOUT         30 min(s)
LIST_COUNT          0 [All Rows]
LOG                 OFF
MARKUP              RAW
PROMPT              SQL>
SCHEMA              SEABASE
SERVER              sqws135.houston.host.com:37800
SQLTERMINATOR       ;
STATISTICS          OFF
TIME                OFF
TIMING              OFF
USER                user1

SQL>
```

8.61. SHOW SQLPROMPT Command

The SHOW SQLPROMPT command displays the value of the SQL prompt for the current TrafCI session.

8.61.1. Syntax

```
SHOW SQLPROMPT
```

8.61.2. Considerations

- You must enter the command on one line. The command does not require an SQL terminator.
- If the SET TIMING command is set to ON, the elapsed time information is displayed.

8.61.3. Example

- This command shows that the SQL prompt for the current session is SQL>:

```
SQL> SHOW SQLPROMPT
SQLPROMPT SQL>
```

8.62. SHOW SQLTERMINATOR Command

The SHOW SQLTERMINATOR command displays the SQL statement terminator of the current TrafCI session.

8.62.1. Syntax

```
SHOW SQLTERMINATOR
```

8.62.2. Considerations

- You must enter the command on one line. The command does not require an SQL terminator.
- If the SET TIMING command is set to ON, the elapsed time information is displayed.

8.62.3. Example

- This command shows that the SQL terminator for the current session is a period (.):

```
SQL> SHOW SQLTERMINATOR

SQLTERMINATOR .
```

For more information, see Set and Show the SQL Terminator.

8.63. SHOW STATISTICS Command

The `SHOW STATISTICS` command displays if statistics has been enabled or disabled for the current session.

8.63.1. Syntax

```
SHOW STATISTICS
```

8.63.2. Considerations

- You must enter the command on one line. The command does not require an SQL terminator.
- If the `SET TIMING` command is set to `ON`, the elapsed time information is displayed.

8.63.3. Example

- This command shows `SHOW STATISTICS` disabled and then enabled:

```
SQL> SHOW STATISTICS

STATISTICS OFF

SQL> SET STATISTICS ON
SQL> SHOW STATISTICS

STATISTICS ON
```

8.64. SHOW TIME Command

The `SHOW TIME` command displays whether the setting for the local time in the interface prompt is `ON` or `OFF`.

8.64.1. Syntax

```
SHOW TIME
```

8.64.2. Considerations

- You must enter the command on one line. The command does not require an SQL terminator.
- If the `SET TIMING` command is set to `ON`, the elapsed time information is displayed.

8.64.3. Example

- This command shows that the setting for the local time in the SQL prompt is `OFF`:

```
SQL> SHOW TIME

TIME OFF
```

8.65. SHOW TIMING Command

The `SHOW TIMING` command displays whether the setting for the elapsed time is `ON` or `OFF`.

8.65.1. Syntax

```
SHOW TIMING
```

8.65.2. Considerations

- You must enter the command on one line. The command does not require an SQL terminator.
- If the `SET TIMING` command is set to `ON`, the elapsed time information is displayed.

8.65.3. Example

- This command displays the elapsed time information because the `SET TIMING` command is enabled:

```
SQL> SET TIMING ON
SQL> SHOW TIME

TIME OFF

Elapsed :00:00:00.000
```

8.66. SPOOL Command

The SPOOL command logs the entered commands and their output from TrafCI to a log file.

8.66.1. Syntax

```
SPOOL { ON [ CLEAR, QUIET, CMDTEXT { ON | OFF } ]
      | log-file [ CLEAR, QUIET, CMDTEXT { ON | OFF } ]
      | OFF
      }
```

- `ON`

  starts the logging process and records information in the `sqlspool.lst` file in the `bin directory.

- `ON CLEAR`

  instructs TrafCI to clear the contents of the `sqlspool.lst` file before logging new information to the file.

- `QUIET`

  specifies that the command text is displayed on the screen, but the results of the command are written only to the log file and not to the screen.

- `CMDTEXT ON`

  specifies that the command text and the log header are displayed in the log file.

- `CMDTEXT OFF`

  specifies that the command text and the log header are not displayed in the log file.

- *`log-file`*

  is the name of a log file into which TrafCI records the entered commands and their output. If you want the log file to exist outside the local directory where you launch TrafCI (by default, the `bin` directory), then specify the full directory path of the log file. The log file does not need to exist, but the specified directory must exist before you execute the SPOOL command.

- *`log-file`* `CLEAR`

  instructs TrafCI to clear the contents of the specified *`log-file`* before logging new information to the file.

- `OFF`

  stops the logging process.

8.66.2. Considerations

- You must enter the command on one line. The command does not require an SQL terminator.
- Use a unique name for each log file to avoid writing information from different TrafCI sessions into the same log file.

8.66.3. Examples

- This command starts the logging process and records information to the `sqlspool.lst` file in the `bin` directory:

```
SQL> SPOOL ON
```

- This command starts the logging process and appends new information to an existing log file, `persnl_updates.log`, in the local directory (the same directory where you are running TrafCI):

```
SQL> SPOOL persnl_updates.log
```

- This command starts the logging process and appends new information to a log file, `sales_updates.log`, in the specified directory on a Windows workstation:

```
SQL> SPOOL c:\log_files\sales_updates.log
```

- This command starts the logging process and appends new information to a log file, `sales_updates.log`, in the specified directory on a Linux or UNIX workstation:

```
SQL> SPOOL ./log_files/sales_updates.log
```

- This command starts the logging process and clears existing information from the log file before logging new information to the file:

```
SQL> SPOOL persnl_ddl.log CLEAR
```

- This command starts the logging process and records information to the `sqlspool.lst` file in the bin directory:

```
SQL> LOG ON
```

- This command starts the logging process and appends new information to an existing log file, `persnl_updates.log`, in the local directory (the same directory where you are running TrafCI):

```
SQL> LOG persnl_updates.log
```

- This command starts the logging process and appends new information to a log file, `sales_updates.log`, in the specified directory on a Windows workstation:

```
SQL> LOG c:\log_files\sales_updates.log
```

- This command starts the logging process and appends new information to a log file, `sales_updates.log`, in the specified directory on a Linux or UNIX workstation:

```
SQL> LOG ./log_files/sales_updates.log
```

- This command starts the logging process and clears existing information from the log file before logging new information to the file:

```
SQL> LOG persnl_ddl.log CLEAR
```

- This command start the logging process, clears existing information from the log file, and specifies that the command text and log header is not displayed in the log file:

```
SQL> LOG c:\temp\a.txt clear, CMDTEXT OFF
SQL> SELECT * FROM trafodion.toi.job
+>;

JOBCODE JOBDESC
------- ------------------
    100 MANAGER
    450 PROGRAMMER
    900 SECRETARY
    300 SALESREP
    500 ACCOUNTANT
    400 SYSTEM ANALYST
    250  ASSEMBLER
    420 ENGINEER
    600 ADMINISTRATOR
    200 PRODUCTION SUPV

--- 10 row(s) selected.

SQL> LOG OFF
```

Output of `c:\temp\a.txt`

```
JOBCODE JOBDESC
------- ------------------
     100 MANAGER
     450 PROGRAMMER 900 SECRETARY
     300 SALESREP
     500 ACCOUNTANT
     400 SYSTEM ANALYST
     250 ASSEMBLER
     420 ENGINEER
     600 ADMINISTRATOR
     200 PRODUCTION SUPV

--- 10 row(s) selected
```

- This command start the logging process, clears existing information from the log file, and specifies that no output appears on the console window:

```
SQL> LOG c:\temp\b.txt CLEAR, CMDTEXT OFF, QUIET
SQL> SELECT *
+>FROM trafodion.toi.job;
SQL> LOG OFF
```

Output of `c:\temp\b.txt`

```
====================
JOBCODE JOBDESC
------- ------------------
     100 MANAGER
     450 PROGRAMMER
     900 SECRETARY
     300 SALESREP
     500 ACCOUNTANT
     400 SYSTEM ANALYST
     250 ASSEMBLER
     420 ENGINEER
     600 ADMINISTRATOR
     200 PRODUCTION SUPV

--- 10 row(s) selected
```

- This command stops the logging process:

```
SQL> LOG OFF
```

For more information, see Log Output.

8.67. VERSION Command

The `VERSION` command displays the build versions of the Trafodion database, Trafodion Connectivity Service, Trafodion JDBC Type 4 Driver, and TrafCI.

8.67.1. Syntax

```
VERSION
```

8.67.2. Considerations

You must enter the command on one line. The command does not require an SQL terminator.

8.67.3. Example

- This command shows versions of the Trafodion database, Trafodion Connectivity Service, Trafodion JDBC Type 4 Driver, and TrafCI:

```
SQL> VERSION

Trafodion Platform              : Release 0.8.0
Trafodion Connectivity Services : Version 1.0.0 Release 0.8.0
Trafodion JDBC Type 4 Driver    : Traf_JDBC_Type4_Build_40646)
Trafodion Command Interface     : TrafCI_Build_40646

SQL>
```

- If TrafCI is started with the -noconnect parameter, the VERSION command displays only TrafCI and the Trafodion JDBC Type 4 Driver versions.

```
C:\Program Files (x86)\Apache Software Foundation\Trafodion Command Interface\bin>
TRAFCI -noconnect

Welcome to Trafodion Command Interface
Copyright(C) 2013-2105 Apache Software Foundation

SQL> VERSION

Trafodion Platform              : Information not available.
Trafodion Connectivity Services : Information not available.
Trafodion JDBC Type 4 Driver    : Traf_JDBC_Type4_Build_40646
Trafodion Command Interface     : TrafCI_Build_40646
```