Preface

Purpose

This book provides information about Basic Teradata Query (BTEQ), which is a Teradata® Tools and Utilities product. Teradata Tools and Utilities is a group of products designed to work with Teradata Database.

Teradata BTEQ software is a general-purpose, command-based program that allows users on a workstation to communicate with one or more Teradata Database systems, and to format reports for both print and screen output.

Audience

This book is intended for use by:

- Anyone who uses BTEQ to access Teradata Database on a regular basis
- System administrators
- Database administrators
- Anyone responsible for creating and maintaining operational scripts and macros.

Supported Releases

This book supports the following releases:

- Teradata Database 13.10
- Teradata Tools and Utilities 13.10
- BTEQ 13.10

**Note:** See "Determining the Current Version of BTEQ" on page 24 to verify the Basic Teradata Query version number.

To locate detailed supported-release information:

2. Click **General Search** under **Online Publications**.
3. Type 3119 in the **Publication Product ID** box.
4. Under **Sort By**, select **Date**.
5. Click **Search**.
Open the version of the *Teradata Tools and Utilities Supported Platforms and Product Versions* spreadsheet associated with this release.

The spreadsheet includes supported Teradata Database versions, platforms, and product release numbers.

## Prerequisites

The following prerequisite knowledge is required for this product:

- Because much of this book is devoted to client channels, networks, and channel- and network-attached computers, you should be familiar with the operations of your channel- or network-attached clients and the systems installed on them.

## Changes to This Book

The following changes were made to this book in support of the current release. Changes are marked with change bars. For a complete list of changes to the product, see the *Release Definition* associated with this release.

<table>
<thead>
<tr>
<th>Date and Release</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>February 2010</td>
<td>• Support on the MP-RAS and z/VM operating systems has been discontinued.</td>
</tr>
<tr>
<td>Release 13.10</td>
<td>• Updated syntax diagrams for LOGON and LOGOFF</td>
</tr>
<tr>
<td></td>
<td>• Added an example of how to use .OS to change the console window title when using Windows BTEQ.</td>
</tr>
<tr>
<td></td>
<td>• File names must be enclosed in quotation marks when text follows the file name.</td>
</tr>
<tr>
<td></td>
<td>• Added a table to the REPORTALIGN command to help explain the effect of the command.</td>
</tr>
<tr>
<td></td>
<td>• Encoding controls for Unicode sessions have been added.</td>
</tr>
<tr>
<td></td>
<td>• The effect of EXPORT RESET on WIDTH has been clarified</td>
</tr>
<tr>
<td></td>
<td>• Support for IPv6 has been added.</td>
</tr>
<tr>
<td></td>
<td>• An example of changing the window title has been included.</td>
</tr>
<tr>
<td></td>
<td>• An example of suppressing title text and dashes for an exported report has been included.</td>
</tr>
</tbody>
</table>
Additional Information

Additional information that supports this product and Teradata Tools and Utilities is available at the web sites listed in the table that follows. In the table, mmyyx represents the publication date of a manual, where mm is the month, y is the last digit of the year, and x is an internal publication code. Match the mmyy of a related publication to the date on the cover of this book. This ensures that the publication selected supports the same release.
<table>
<thead>
<tr>
<th>Type of Information</th>
<th>Description</th>
<th>Access to Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release overview</td>
<td>Use the Release Definition for the following information:</td>
<td>1  Go to <a href="http://www.info.teradata.com/">http://www.info.teradata.com/</a>.</td>
</tr>
<tr>
<td>Late information</td>
<td>• Overview of all of the products in the release</td>
<td>2  Click <strong>General Search</strong> under <strong>Online Publications</strong>.</td>
</tr>
<tr>
<td></td>
<td>• Information received too late to be included in the manuals</td>
<td>3  Type 2029 in the <strong>Publication Product ID</strong> box.</td>
</tr>
<tr>
<td></td>
<td>• Operating systems and Teradata Database versions that are certified to work with each product</td>
<td>4  Click <strong>Search</strong>.</td>
</tr>
<tr>
<td></td>
<td>• Version numbers of each product and the documentation for each product</td>
<td>5  Select the appropriate Release Definition from the search results.</td>
</tr>
<tr>
<td></td>
<td>• Information about available training and the support center</td>
<td></td>
</tr>
<tr>
<td>Additional product</td>
<td>Use the Teradata Information Products website to view or download specific manuals that supply related or additional information to this manual.</td>
<td>1  Go to <a href="http://www.info.teradata.com/">http://www.info.teradata.com/</a>.</td>
</tr>
<tr>
<td>information</td>
<td></td>
<td>2  Click <strong>Data Warehousing</strong> under <strong>Online Publications, Browse by Category</strong>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3  Do one of the following:</td>
</tr>
<tr>
<td></td>
<td>• For a list of Teradata Tools and Utilities documents, click <strong>Teradata Tools and Utilities</strong>, and then select an item under <strong>Releases</strong> or <strong>Products</strong>.</td>
<td>4  Select a link to any of the data warehousing publications categories listed.</td>
</tr>
<tr>
<td></td>
<td>• Specific books related to BTEQ are as follows:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>Teradata Call-Level Interface Version 2 Reference for Channel-Attached Systems</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>Teradata Call-Level Interface Version 2 Reference for Network-Attached Systems</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>SQL Data Definition Language</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>SQL Fundamentals</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>SQL External Routine Programming</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>Messages</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>SQL Request and Transaction Processing</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>SQL Data Manipulation Language</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>Teradata Tools and Utilities Command Summary</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>SQL Functions, Operators, Expressions, and Predicates</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>Security Administration</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>International Character Set Support</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>Teradata Director Program Reference</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• <strong>Teradata Tools and Utilities Installation Guide for IBM z/OS</strong></td>
<td></td>
</tr>
<tr>
<td>Type of Information</td>
<td>Description</td>
<td>Access to Information</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| CD-ROM images                | Access a link to a downloadable CD-ROM image of all customer documentation for this release. Customers are authorized to create CD-ROMs for their use from this image. | 1 Go to [http://www.info.teradata.com/](http://www.info.teradata.com/).  
2 Click Data Warehousing under Online Publications, Browse by Category.  
3 Click CD-ROM List and Images. |
| Ordering information for manuals | Use the Teradata Information Products website to order printed versions of manuals.          | 1 Go to [http://www.info.teradata.com/](http://www.info.teradata.com/).  
2 Click How to Order under Print & CD Publications.  
3 Follow the ordering instructions. |
| General information about Teradata | The Teradata home page provides links to numerous sources of information about Teradata. Links include:  
• Executive reports, case studies of customer experiences with Teradata, and thought leadership  
• Technical information, solutions, and expert advice  
• Press releases, mentions, and media resources | 1 Go to Teradata.com.  
2 Select a link. |
# Table of Contents

## Preface

- Purpose ................................................................. 3
- Audience ............................................................... 3
- Supported Releases .................................................. 3
- Prerequisites .......................................................... 4
- Changes to This Book ................................................ 4
- Additional Information ............................................... 5

## Chapter 1: Introduction to BTEQ ........................................... 23

- Overview of BTEQ .................................................... 23
- Determining the Current Version of BTEQ ....................... 24
- BTEQ Operation in the Teradata Client-Server Environment ... 24
- BTEQ Communication ............................................... 25
- Channel-Attached Communication Diagram .................... 25
- Network-Attached Communication Diagram ..................... 26
- BTEQ Process .......................................................... 26
- Call-Level Interface ................................................ 27
- Teradata Director Program (TDP) (or MTDP) .................... 27
- Teradata Server ...................................................... 27
- Operating Environments ............................................ 27
- Command Set .......................................................... 28
- Session Control Commands ........................................ 28
- Controlling SQL ....................................................... 30
- File Control Commands ............................................. 30
- Sequence Control Commands ...................................... 31
- Format Control Commands ......................................... 32
Chapter 2: Starting and Exiting BTEQ

Obtaining System Access ........................................... 35
  System Access Requirements .................................. 35
Starting a BTEQ Session ........................................... 36
  Starting BTEQ ....................................................... 36
  Encoding Controls for Unicode Sessions ...................... 37
Command Line Options ............................................ 37
Automatic File Allocation .......................................... 39
LOGON Command .................................................. 39
  LOGON Command Elements .................................... 39
  Specifying the tdpid ............................................ 40
  Specifying Your userid ........................................ 40
  Specifying Your password .................................... 40
  Specifying Your acctid ........................................ 40
Logging on to Teradata Database ............................... 40
  Logging On in Interactive Mode .............................. 41
  Logging On in Batch Mode .................................... 41
  Bypassing Logon Prompts ..................................... 41
  Successful Logon Message .................................... 41
  Failed Logon Message ........................................... 42
  Retrying a Failed Logon Attempt ............................. 42
LOGOFF Command ................................................ 42
EXIT and QUIT Commands ........................................ 43
Error Information Options ....................................... 43
Changing Your Password ........................................ 43
  When Your Password Expires .................................. 43
  Replacing Your Expired Password ............................ 44
  Replacing Your Password at Any Time ....................... 44
Specifying a Different Teradata Server ....................... 44
  System Default tdpid .......................................... 44
  Changing the tdpid ............................................. 44
  Changing the Server for a Single Logon ..................... 45
  Changing the Server for the Current Session ............... 45
Specifying Multiple Sessions ................................... 45
  Maximum Number of Sessions in a Set ...................... 45
  Setting the Number of Sessions ............................. 45
Getting Help ........................................................ 46
  Syntax Conventions ............................................ 46
# Chapter 3: Using BTEQ

Using Teradata SQL in BTEQ ........................................ 47
  Request Types ........................................ 47
  Multi-Statement Processing ................................ 49
  Negating Requests ...................................... 49
  Continuing SQL ......................................... 50
Using Comments in a BTEQ Session .............................. 50
  BTEQ Comment Elements ................................ 50
  ANSI Comment Elements ................................ 51
  Mixing Comments ....................................... 51
  Concatenating BTEQ Comments ........................... 51
  Illegal Comments ....................................... 51
Entering IPv4 and IPv6 Network Addresses ........................ 52
UNIX Signals ....................................................... 52
Running BTEQ Batch Jobs ....................................... 52
  Running Under z/OS ..................................... 52
  Running Other Client Systems ........................... 53
  Switching from Interactive to Batch Mode ............... 54
Defining Input and Output Files for BTEQ in Batch Mode ...... 54
  File Types .................................................. 54
  Defining Input and Output Files ......................... 54
  Defining Additional Files ................................ 55
  Specifying a Standard Input File ......................... 55
  Using a Here Document for Standard Input Files ....... 56
  Specifying a Standard Output File ....................... 56
  Specifying an Import File ................................ 57
  Specifying Files for User-Defined Functions ............. 58
  Specifying an Export File ................................ 59
  Specifying a Run File .................................... 59
  Specifying a Standard Error Output File ................ 59
Using Scripts, Run Files, Macros, and Stored Procedures ....... 61
  Line Numbers in Files .................................. 62
  Command Execution Order ................................ 62
  Creating and Using Scripts ............................... 62
  Creating and Using Run Files ............................. 63
  Revising Scripts and Run Files ........................... 65
  Creating and Using Macros ............................... 66

Notes on Command Input ........................................ 46

Basic Teradata Query Reference 11
# Table of Contents

- Creating and Using Stored Procedures .................................................. 72
- Testing and Branching ........................................................................... 75
  - Testing and Branching Commands ....................................................... 75
  - Testing Status Values ....................................................................... 77
  - Testing Conditions ............................................................................ 77
  - Producing Conditional Messages ....................................................... 78
- Using the REPEAT Command ................................................................. 78
  - Updating Rows Using a Single Request ............................................. 78
  - Updating Rows Using Multiple Requests ......................................... 78
  - Updating Rows by Importing a File .................................................. 79
  - Using the REPEAT Command when Importing a File ....................... 79
  - Using the = Command When Importing a File ................................. 79
  - Using REPEAT with Multiple Sessions ............................................ 80
- Interrupting and Aborting Requests .................................................... 81
  - Using the Break Key on Workstations ............................................. 81
  - Using the Break Key on z/OS .............................................................. 82
  - Aborting Requests with the ABORT Command .................................. 83
- Entering Operating System Commands from a BTEQ Session .......... 84
  - Entering UNIX Commands ............................................................... 84
  - Entering Windows Commands ......................................................... 85
- Handling Errors ..................................................................................... 85
  - Teradata Database Error Codes ....................................................... 85
  - Possible Data-Loss Condition ............................................................ 85
  - BTEQ Return Codes ......................................................................... 86
  - Error Severity Levels ....................................................................... 90
  - Maximum Error Level ..................................................................... 90
  - Stored Procedure Compilation ....................................................... 91
  - BTEQ Error Output Files ................................................................ 91
- I/O Errors and Abends ........................................................................... 92
  - Channel-Attached Clients: ................................................................. 92
  - BTEQ Large File (LF) Support (for files >2GB) ................................ 93

---

## Chapter 4: BTEQ Output

- Formatting BTEQ Output: Default Specifications .................................. 95
  - How Defaults Are Inherited ............................................................... 95
  - Viewing the Current Format Specifications .................................... 95
  - Specifying the Default Format Specifications .................................. 95
  - Changing the Screen Width .............................................................. 96
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Changing the Null Specification</td>
<td>96</td>
</tr>
<tr>
<td>Disabling the Echo Required Function</td>
<td>96</td>
</tr>
<tr>
<td>Disabling the Retry Function</td>
<td>97</td>
</tr>
<tr>
<td>When an SQL Statement Fails in Teradata Mode</td>
<td>97</td>
</tr>
<tr>
<td>Specifying Record or Indicator Mode</td>
<td>98</td>
</tr>
<tr>
<td>Numeric Overflow in Reports</td>
<td>99</td>
</tr>
<tr>
<td>Limiting the Amount of Display Data</td>
<td>100</td>
</tr>
<tr>
<td>Specifying Report Format</td>
<td>101</td>
</tr>
<tr>
<td>Vertical Presentation</td>
<td>101</td>
</tr>
<tr>
<td>Horizontal Presentation</td>
<td>102</td>
</tr>
<tr>
<td>Deleting Duplicated Data</td>
<td>102</td>
</tr>
<tr>
<td>Inserting Blank Lines</td>
<td>103</td>
</tr>
<tr>
<td>Inserting a Line of Underline Characters</td>
<td>103</td>
</tr>
<tr>
<td>Specifying Page Breaks</td>
<td>104</td>
</tr>
<tr>
<td>Specifying Column Separations</td>
<td>105</td>
</tr>
<tr>
<td>Splitting Report Lines</td>
<td>105</td>
</tr>
<tr>
<td>Horizontal Presentation</td>
<td>105</td>
</tr>
<tr>
<td>Vertical Presentation</td>
<td>105</td>
</tr>
<tr>
<td>Specifying Side-Titles</td>
<td>106</td>
</tr>
<tr>
<td>Specifying Headers and Footers</td>
<td>107</td>
</tr>
<tr>
<td>BTEQ Commands to Use</td>
<td>107</td>
</tr>
<tr>
<td>Specifying Printer Formats</td>
<td>109</td>
</tr>
<tr>
<td>Specifying the Page Size</td>
<td>109</td>
</tr>
<tr>
<td>Specifying the Form Feed Character</td>
<td>110</td>
</tr>
<tr>
<td>WITH Clause Output</td>
<td>110</td>
</tr>
<tr>
<td>Sequence of Fields</td>
<td>110</td>
</tr>
<tr>
<td>Summary Title Accommodation</td>
<td>111</td>
</tr>
<tr>
<td>Title Phrases</td>
<td>113</td>
</tr>
<tr>
<td>Converting Screen Displays to Reports</td>
<td>115</td>
</tr>
<tr>
<td>Producing a Report</td>
<td>115</td>
</tr>
<tr>
<td>Channel-Attached Example</td>
<td>116</td>
</tr>
<tr>
<td>Identifying the Input File</td>
<td>116</td>
</tr>
<tr>
<td>Creating an Output Log File</td>
<td>116</td>
</tr>
<tr>
<td>Printing the Report File</td>
<td>117</td>
</tr>
<tr>
<td>Resubmitting Output as Input</td>
<td>117</td>
</tr>
<tr>
<td>Procedure For Resubmitting Output as Input</td>
<td>117</td>
</tr>
<tr>
<td>Printing Output on z/OS</td>
<td>119</td>
</tr>
<tr>
<td>Automatically Converted Characters</td>
<td>119</td>
</tr>
</tbody>
</table>
# Chapter 5: BTEQ Commands

BTEQ Command Syntax .................................................. 121
Default BTEQ Values .................................................... 124
BTEQ Command Descriptions ........................................... 125

- ABORT ................................................................. 130
- AUTOKEYRETRIEVE ................................................. 131
- COMPILER .......................................................... 134
- DECIMALDIGITS .................................................... 138
- DEFAULTS ......................................................... 140
- ECHOREQ ............................................................ 145
- ENCRYPTION ....................................................... 146
- ERRORLEVEL ....................................................... 147
- ERROROUT .......................................................... 149
- EXIT ................................................................. 151
- EXPORT ............................................................ 152
- EXPORTJECT ...................................................... 161
- FOLDLINE .......................................................... 163
- FOOTING ............................................................. 169
- FORMAT .............................................................. 174
- FORMCHAR ........................................................ 176
- FULLYEAR .......................................................... 178
- GOTO ................................................................. 179
- HANG ............................................................... 180
- HEADING ............................................................ 181
- HELP BTEQ .......................................................... 186
- HX ................................................................. 187
- IF...THEN... ....................................................... 188
- IMPORT ............................................................ 190
- INDICDATA ........................................................ 197
- LABEL .............................................................. 200
- LARGEDATAMODE ............................................... 201
- LOGMECH and LOGDATA ........................................ 203
- LOGOFF ............................................................. 205
- LOGON ............................................................ 206
- LOGONPROMPT .................................................... 211
- MAXERROR ......................................................... 213
- MESSAGEOUT ..................................................... 214
- NOTIFY ............................................................. 216
- NULL ............................................................... 229
## Table of Contents

- OMIT .................................................. 232
- OS .................................................. 235
- PACK ............................................... 237
- PAGEBREAK ...................................... 238
- PAGELENGTH ................................. 241
- QUIET ............................................ 243
- QUIT .......................................... 245
- RECORDMODE ............................... 246
- REMARK ....................................... 249
- REPEAT ........................................ 252
- REPEATSTOP ................................. 257
- REPORTALIGN ......................... 258
- RETCANCEL ................................ 260
- RETLIMIT ..................................... 262
- RETRY ...................................... 265
- RTITLE ....................................... 267
- RUN ........................................ 271
- SEPARATOR .................................. 273
- SESSION CHARSET ....................... 276
- SESSION RESPBUFLEN .............. 280
- SESSION SQLFLAG .................... 282
- SESSION TRANSACTION .......... 284
- SESSION TWORESPBUFS ........... 286
- SESSIONS .................................. 287
- SHOW CONTROLS ......................... 289
- SHOW ERRORMAP ..................... 295
- SHOW VERSIONS ....................... 297
- SIDETITLES .................................. 299
- SKIPDOUBLE ............................... 302
- SKIPLINE .................................. 304
- SUPPRESS .................................. 307
- TDP .......................................... 310
- TIMEMSG ..................................... 312
- TITLEDASHES ............................ 315
- TSO ........................................ 318
- UNDERLINE ................................ 319
- WIDTH ...................................... 322

## Appendix A:
### How to Read Syntax Diagrams ........................................ 325

Syntax Diagram Conventions ........................................ 325
# Table of Contents

Strings ................................................................. 327
Multiple Legitimate Phrases ........................................... 329
Sample Syntax Diagram ............................................... 330
Diagram Identifier ..................................................... 330

## Appendix B:
Loading and Using the Sample Personnel Database ............... 331

Database Contents ..................................................... 331
Preparing to Use the Sample Database ................................ 333
Loading the Personnel Database ...................................... 334
  Installation Script .................................................... 334
  Copying the Personnel Database .................................... 334
Listing Salary by Location and Department .......................... 337
  BTEQ Input Stream .................................................. 338
  Using Windows or UNIX ................................................ 340
Listing Total Earnings by Department ................................ 340
  BTEQ Input Stream .................................................. 340
  Using Windows or UNIX ................................................ 341
  Using z/OS ........................................................... 341
Extracting Data From Teradata Database ............................. 342
  BTEQ Input Stream .................................................. 342
  Using Windows or UNIX ................................................ 342
  Using z/OS ........................................................... 343
Loading Data into Teradata Database ................................ 343
  BTEQ Input Stream .................................................. 343
  Using Windows or UNIX ................................................ 344
  Using z/OS ........................................................... 344
Using a z/OS Batch Job to Extract and Load Data .................... 345
Listing the Error Log File .............................................. 346

## Appendix C:
Differences Between Channel-Attached and Network-Attached Systems ........................................... 349

Commands and Keywords .............................................. 349
### Appendix D: Kanji, Chinese, Korean, and Unicode Character Set Support

351

Kanji Character Sets for Channel-Attached Systems 351
Kanji Character Sets for Network-Attached Systems 351
   Establishing the Character Set 351
   Description 352
   Message Display 353
SESSION CHARSET Command 353
GRAPHIC, VARGRAPHIC and LONG VARGRAPHIC 353
Chinese and Korean Character Sets 354
   Rules 354
Unicode Character Set 355
   Rules 355

### Appendix E: Using BTEQWIN

357

What is BTEQWIN? 357
BTEQWIN Main Window 357
BTEQWIN Menus 359
   BTEQWIN Menu Bar 359
Displaying Multi-Byte Characters 362

### Glossary

363

### Index

367
Table of Contents
List of Figures

Figure 1: BTEQ Operating in a Channel-Attached System .......................... 25
Figure 2: BTEQ Operating in a Network-Attached System .......................... 26
Figure 3: Coding Controls Effect on BTEQ .......................... 37
Figure 4: Total Salary by Location, Department .......................... 338
Figure 5: Total Earnings by Department .......................... 340
Figure 6: Error Log Listing .......................... 346
Figure 7: Select Session Charset Dialog .......................... 358
Figure 8: BTEQWIN Main Window .......................... 359
List of Tables

Table 1: BTEQ Commands for Teradata SQL Macros ................................. 66
Table 2: Return Code 04 Default Error Message Assignments ...................... 87
Table 3: Return Code 08 Default Error Message Assignments ...................... 87
Table 4: Return Code 12 Default Error Message Assignments ...................... 89
Table 5: BTEQ Command Elements .......................................................... 122
Table 6: Syntax Variable Definitions ....................................................... 123
Table 7: BTEQ Commands and DEFAULTS Relationship .............................. 140
Table 8: Events That Create Notifications ............................................... 218
Table 9: Building Exit Modules Using Different Operating Systems .............. 220
Table 10: Employee Table, Personnel Database ........................................ 331
Table 11: Department Table, Personnel Database ...................................... 332
Table 12: Project Table, Personnel Database ........................................... 332
Table 13: Charges Table, Personnel Database .......................................... 333
List of Tables
CHAPTER 1
Introduction to BTEQ

This chapter introduces BTEQ (pronounced “bee-teek”). This chapter provides the following information:

- Overview of BTEQ
- BTEQ Operation in the Teradata Client-Server Environment
- BTEQ Process
- Operating Environments
- Command Set

Overview of BTEQ

BTEQ is an abbreviation of Basic Teradata Query, which is a general-purpose, command-based program that allows users on a workstation to communicate with one or more Teradata Database systems, and to format reports for both print and screen output. Using BTEQ you can submit SQL queries to Teradata Database. BTEQ formats the results and returns them to the screen, a file, or to a designated printer.

Environment

You install and run BTEQ on the client portion of either a channel-attached or a network-attached system.

BTEQ Sessions

A BTEQ session provides a quick and easy way to access Teradata Database. In a BTEQ session, you can do the following:

- Establish one or more Teradata Database sessions to submit single- and multi-statement SQL requests using either Teradata Mode or ANSI standard transaction semantics
- Use Teradata SQL statements to view, add, modify, and delete data
- Use formatting commands to produce field-mode-based columnar reports
- Specify operating system commands
- Create database objects, such as user-defined functions and stored procedures, that require transfer of source files
BTEQ Session Modes

BTEQ operates in both batch and interactive modes:

<table>
<thead>
<tr>
<th>In...</th>
<th>You...</th>
</tr>
</thead>
<tbody>
<tr>
<td>interactive mode</td>
<td>start a BTEQ session, and submit requests to the database as needed</td>
</tr>
<tr>
<td>batch mode</td>
<td>prepare scripts or macros and submit them to BTEQ for processing</td>
</tr>
</tbody>
</table>

Determining the Current Version of BTEQ

Windows

To determine the currently installed version of the BTEQ on Windows, select Start > Control Panel > Add or Remove Programs and verify the driver version number.

UNIX and Linux

To determine the currently installed version of the BTEQ and Linux, use one of the following commands.

On Sun Solaris:

pkginfo -l BTEQ

On IBM AIX:

lslpp -l BTEQ

On HP-UX:

swlist BTEQ

On Red Hat Linux:

rpm -q <rpmname>

On SUSE Linux:

rpm -ql <rpmname>

On zLinux (z/OS):

rpm -ql <rpmname>

BTEQ Operation in the Teradata Client-Server Environment

This section describes how BTEQ operates in the Teradata client-server environment, and briefly describes the logical components of the environment that enable BTEQ to communicate with Teradata Database.
BTEQ Communication

BTEQ resides on the client portion of either a channel-attached or network-attached system, and communicates with one or more Teradata Database systems residing on the server. The client system communicates with Teradata Database as described in the following table:

<table>
<thead>
<tr>
<th>If your client system is...</th>
<th>Then communication occurs over...</th>
</tr>
</thead>
<tbody>
<tr>
<td>channel-attached</td>
<td>a high-speed I/O channel.</td>
</tr>
<tr>
<td>network-attached</td>
<td>a local area network.</td>
</tr>
</tbody>
</table>

Channel-Attached Communication Diagram

Figure 1: BTEQ Operating in a Channel-Attached System
BTEQ Process

This section describes the BTEQ process.

**Note:** For a more detailed description of Teradata Database system components and their roles in processing input and output, see *Introduction to Teradata*.

1. Issue BTEQ commands and submit SQL requests in a batch- or interactive-mode BTEQ session.

2. The Call-Level Interface packages the input and forwards it to the Teradata Director Program (TDP) (in channel-attached systems) or the Micro Teradata Director Program (MTDP) (in network-attached systems).

3. The TDP (or MTDP) establishes a session and manages communication between the CLI and the Teradata server.

4. The Teradata Server parses, interprets, plans access for, and dispatches the request to Teradata Database.

5. Teradata Database processes the request and sends a response message back to the TDP (or MTDP).

6. The TDP (or MTDP) unpackages the response and makes it available to the CLI.
The CLI makes the response available to BTEQ, which routes the output to the specified terminal, file, or printer.

**Call-Level Interface**

The Call-Level Interface packages input to Teradata Database, and unpackages responses from Teradata Database.

**Note:** All references to the Call-Level Interface in this manual refer to CLI version 2 (CLIv2).

For information on the Call-Level Interface, see the appropriate document for your system configuration:

- *Teradata Call-Level Interface Version 2 Reference for Channel-Attached Systems*
- *Teradata Call-Level Interface Version 2 Reference for Network-Attached Systems*

**Teradata Director Program (TDP) (or MTDP)**

The TDP manages data communications, including:

- logging
- recovery
- restarts
- physical I/O
- session and security management

in channel-attached systems.

The Micro Teradata Director Program (MTDP) performs the same functions as the TDP in network-attached systems.

**Teradata Server**

The Teradata server is a collection of software that does the following for one or more Teradata Database systems:

- parses and dispatches SQL requests
- controls sessions and balances the communications load
- manages the database
- includes the Teradata file system and Parallel Database Extensions (PDE)

**Operating Environments**

To locate detailed supported-release information:

2. Navigate to **General Search>Publication Product ID**.
3. Enter *[3119]*.
The BTEQ command set can be categorized as:
- Session control commands
- File control commands
- Sequence control commands
- Format control commands

BTEQ commands enable you to perform data control functions; they do not act directly on data.

BTEQ commands can be categorized into four functional groups, as described in the following table:

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session control</td>
<td>begin and end BTEQ sessions, and control session characteristics.</td>
</tr>
<tr>
<td>File control</td>
<td>specify input and output formats and identify information sources and</td>
</tr>
<tr>
<td></td>
<td>destinations.</td>
</tr>
<tr>
<td>Sequence control</td>
<td>control the sequence in which other BTEQ commands and Teradata</td>
</tr>
<tr>
<td></td>
<td>SQL statements will be executed within scripts.</td>
</tr>
<tr>
<td>Format control</td>
<td>control the format of screen and printer output.</td>
</tr>
</tbody>
</table>

### Session Control Commands

Use the following BTEQ commands to begin, control, and end sessions:

- `ABORT`          SESSION CHARSET
- `COMPILE`        SESSION RESPBUFLEN
- `DECIMALDIGITS`  SESSION SQLFLAG
- `DEFAULTS`       SESSION TRANSACTION
- `EXIT`           SESSION TWORESPBUFS
- `HALT EXECUTION`  SESSIONS
- `LOGOFF`         SHOW CONTROLS
- `LOGON`          SHOW VERSIONS
- `LOGONPROMPT`    TDP
- `QUIT`
## Controlling Sessions

<table>
<thead>
<tr>
<th>If you want to…</th>
<th>Use this command…</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Start a Session</strong></td>
<td></td>
</tr>
<tr>
<td>start a BTEQ session,</td>
<td>LOGON</td>
</tr>
<tr>
<td>specify the number of sessions to use with the next LOGON command,</td>
<td>SESSIONS</td>
</tr>
<tr>
<td>specify the Teradata server for subsequent logons during the current session,</td>
<td>TDP</td>
</tr>
<tr>
<td>bypass the warnings related to conventional LOGON command use,</td>
<td>LOGONPROMPT</td>
</tr>
<tr>
<td><strong>End a Session</strong></td>
<td></td>
</tr>
<tr>
<td>end the current sessions without exiting BTEQ,</td>
<td>LOGOFF</td>
</tr>
<tr>
<td>end the current sessions and exit BTEQ,</td>
<td>EXIT or QUIT</td>
</tr>
<tr>
<td>abort any active requests and transactions without exiting BTEQ,</td>
<td>ABORT</td>
</tr>
<tr>
<td>abort any active requests and transactions and exit BTEQ,</td>
<td>HALT EXECUTION</td>
</tr>
<tr>
<td><strong>Display Information about the Session</strong></td>
<td></td>
</tr>
<tr>
<td>display the current configuration of the BTEQ control command options,</td>
<td>SHOW CONTROLS</td>
</tr>
<tr>
<td>display the BTEQ version number, module revision numbers, and linking date,</td>
<td>SHOW VERSIONS</td>
</tr>
<tr>
<td><strong>Specify Session Characteristics</strong></td>
<td></td>
</tr>
<tr>
<td>specify the name of a character set for the current session,</td>
<td>SESSION CHARSET</td>
</tr>
<tr>
<td>specify the disposition of warnings issued in response to violations of ANSI-</td>
<td>SESSION SQLFLAG</td>
</tr>
<tr>
<td>specify whether transaction boundaries are determined by Teradata SQL</td>
<td>SESSION TRANSACTION</td>
</tr>
<tr>
<td>specify whether CLI double-buffering is used,</td>
<td>SESSION TWORESPBUFS</td>
</tr>
<tr>
<td>create or replace a Teradata stored procedure</td>
<td>COMPILe</td>
</tr>
<tr>
<td>override the buffer length specified in <em>resp_buf_len</em></td>
<td>SESSION RESPBUFLEN</td>
</tr>
</tbody>
</table>
Controlling SQL

Use the following BTEQ commands to control SQL:

<table>
<thead>
<tr>
<th>If you want to…</th>
<th>Use this command…</th>
</tr>
</thead>
<tbody>
<tr>
<td>override the precision specified by a CLI System Parameter Block (SPB) max_decimal_returned entry, or if that entry does not exist, to indicate what the precision should be for decimal values associated with subsequently issued SQL requests for non-Field Mode responses.</td>
<td>DECIMALDIGITS</td>
</tr>
<tr>
<td>enable users to specify whether the values of any fields associated with Identity Data are returned in response to an SQL Insert operation.</td>
<td>AUTOKEYRETRIEVE</td>
</tr>
</tbody>
</table>

File Control Commands

Use the following BTEQ commands to specify the format of incoming and outgoing information and identify the source and destination of input and output streams:

<table>
<thead>
<tr>
<th>If you want to…</th>
<th>Use this command…</th>
</tr>
</thead>
<tbody>
<tr>
<td>execute a z/OS TSO command from within the BTEQ environment,</td>
<td>TSO</td>
</tr>
</tbody>
</table>

Controlling Input Files

See Chapter 5: “BTEQ Commands” for complete syntax and command description.
Chapter 1: Introduction to BTEQ

Command Set

See Chapter 5: “BTEQ Commands” for the complete syntax and description of each command.

### Controlling Output Files

<table>
<thead>
<tr>
<th>If you want to...</th>
<th>Use this command...</th>
</tr>
</thead>
<tbody>
<tr>
<td>execute a Windows or UNIX command from within the BTEQ environment,</td>
<td>OS</td>
</tr>
<tr>
<td>execute Teradata SQL requests and BTEQ commands from a specified run file,</td>
<td>RUN</td>
</tr>
<tr>
<td>submit the next request a specified number of times,</td>
<td>REPEAT</td>
</tr>
</tbody>
</table>

### Sequence Control Commands

Commands that affect the sequence in which other commands and requests are submitted are most useful in scripts. Use these BTEQ commands to control the sequence in which BTEQ executes commands:

<table>
<thead>
<tr>
<th>If you want to...</th>
<th>Use this command...</th>
</tr>
</thead>
<tbody>
<tr>
<td>abort any active requests and transactions and exit BTEQ,</td>
<td>HALT EXECUTION</td>
</tr>
<tr>
<td>enable or inhibit the page-oriented format command options,</td>
<td>FORMAT</td>
</tr>
<tr>
<td>open a file with a specific format to transfer information from Teradata Database,</td>
<td>EXPORT</td>
</tr>
<tr>
<td>specify the response mode (Field Mode, Indicator Mode, Record Mode, or Multipart Indicator Mode) for data selected from Teradata Database,</td>
<td>INDICDATA and/or RECORDMODE</td>
</tr>
</tbody>
</table>

### Controlling Command Sequences

<table>
<thead>
<tr>
<th>If you want to...</th>
<th>Use this command...</th>
</tr>
</thead>
<tbody>
<tr>
<td>pause BTEQ processing for a specified period of time,</td>
<td>HANG</td>
</tr>
</tbody>
</table>

See Chapter 5: “BTEQ Commands” for the complete syntax and description of each command.
Format Control Commands

Use the following BTEQ commands to specify the way BTEQ presents information for screen-oriented and printer/printer-file oriented output:

<table>
<thead>
<tr>
<th>DEFAULTS</th>
<th>OMIT</th>
<th>SEPARATOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECHOREQ</td>
<td>PAGEBREAK</td>
<td>SHOW CONTROLS</td>
</tr>
<tr>
<td>EXPORT</td>
<td>PAGELENGTH</td>
<td>SIDETITLES</td>
</tr>
<tr>
<td>FOLDLINE</td>
<td>QUIET</td>
<td>SKIPDOUBLE</td>
</tr>
<tr>
<td>FOOTING</td>
<td>RECORDMODE</td>
<td>SKIPLINE</td>
</tr>
<tr>
<td>FORMAT</td>
<td>REPORTALIGN</td>
<td>SUPPRESS</td>
</tr>
<tr>
<td>HEADING</td>
<td>RETCANCEL</td>
<td>TITLEDASHES</td>
</tr>
<tr>
<td>IMPORT</td>
<td>RETLIMIT</td>
<td>UNDERLINE</td>
</tr>
<tr>
<td>INDICDATA</td>
<td>RETRY</td>
<td>WIDTH</td>
</tr>
<tr>
<td>NULL</td>
<td>RTITLE</td>
<td></td>
</tr>
</tbody>
</table>

Controlling Formats

<table>
<thead>
<tr>
<th>If you want to...</th>
<th>Use this command...</th>
</tr>
</thead>
<tbody>
<tr>
<td>cancel a request when the value specified by the RETLIMIT command ROWS option is exceeded.</td>
<td>RETCANCEL</td>
</tr>
<tr>
<td>control the spacing of UTF8 report mode data,</td>
<td>REPORTALIGN</td>
</tr>
<tr>
<td>display a row of dash characters before each report line summarized by a WITH clause,</td>
<td>TITLEDASHES</td>
</tr>
<tr>
<td>display a row of dash characters whenever the value of a specified column changes,</td>
<td>UNDERLINE</td>
</tr>
<tr>
<td>enable the echo required function that returns a copy of each Teradata SQL request and BTEQ command to the standard output stream,</td>
<td>ECHOREQ</td>
</tr>
<tr>
<td>enable the page-oriented format command options,</td>
<td>FORMAT</td>
</tr>
<tr>
<td>insert a blank line in a report whenever the value of a specified column changes,</td>
<td>SKIPLINE</td>
</tr>
<tr>
<td>insert two blank lines in a report whenever the value of a specified column changes,</td>
<td>SKIPDOUBLE</td>
</tr>
<tr>
<td>insert three blank lines in a report whenever the value of a specified column changes,</td>
<td>SKIPLINE, then SKIPDOUBLE</td>
</tr>
<tr>
<td>limit BTEQ output displays to error messages and request processing statistics,</td>
<td>QUIET</td>
</tr>
<tr>
<td>open a file with a specific format to transfer information to Teradata Database,</td>
<td>IMPORT</td>
</tr>
<tr>
<td>open a file with a specific format to transfer information from Teradata Database,</td>
<td>EXPORT</td>
</tr>
<tr>
<td>If you want to…</td>
<td>Use this command…</td>
</tr>
<tr>
<td>--------------------------------------------------------------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>position summary titles to the left of the summary lines in a report,</td>
<td>SIDETITLES</td>
</tr>
<tr>
<td>replace all consecutively repeated values with all-blank character strings,</td>
<td>SUPPRESS</td>
</tr>
<tr>
<td>reset the BTEQ format command options to their default configurations,</td>
<td>DEFAULTS</td>
</tr>
<tr>
<td>resubmit requests that fail under certain error conditions,</td>
<td>RETRY</td>
</tr>
<tr>
<td>specify a character or character string to represent null field values returned from Teradata Database,</td>
<td>NULL</td>
</tr>
<tr>
<td>specify a character string or width (in blank characters) to separate columns of a report,</td>
<td>SEPARATOR</td>
</tr>
<tr>
<td>specify a footer to appear at the bottom of every page of a report,</td>
<td>FOOTING</td>
</tr>
<tr>
<td>specify a header to appear at the top of every page of a report,</td>
<td>HEADING or RTITLE</td>
</tr>
<tr>
<td>specify the response mode (Field Mode, Indicator Mode, or Record Mode) for data selected from Teradata Database,</td>
<td>INDICDATA and/or RECORDMODE</td>
</tr>
<tr>
<td>specify the maximum number of rows and/or columns displayed or written in response to a Teradata SQL request,</td>
<td>RETLIMIT</td>
</tr>
<tr>
<td>specify the page length of printed reports, in lines per page,</td>
<td>PAGELENGTH</td>
</tr>
<tr>
<td>specify the width of screen displays and printed reports, in characters per line,</td>
<td>WIDTH</td>
</tr>
<tr>
<td>split (fold) each line of a report into two or more lines,</td>
<td>FOLDLINE</td>
</tr>
</tbody>
</table>

See Chapter 5: “BTEQ Commands” for the complete syntax and description of each command.
This chapter guides you through the process of

- starting, and
- stopping BTEQ sessions

in both

- interactive, and
- batch mode.

It also describes some special session management functions.

The table below describes the process of starting and stopping a BTEQ session.

1. Obtain client (BTEQ) and server (Teradata Database) system access. See “Obtaining System Access.”
2. Invoke BTEQ on the client system. See “Starting a BTEQ Session.”
3. Log on to Teradata Database from the client system. See “Logging on to Teradata Database.”
4. Log off from Teradata Database. See “Logging Off Teradata Database / Exiting Database.”
5. Exit BTEQ.

### Obtaining System Access

To use BTEQ, you must be able to access both of the following:

- the client system running BTEQ
- Teradata Database

### System Access Requirements

The following table describes typical system access requirements:

<table>
<thead>
<tr>
<th>To access</th>
<th>You need</th>
<th>You may also need</th>
</tr>
</thead>
<tbody>
<tr>
<td>the client system,</td>
<td>a valid <em>userid</em> and password.</td>
<td>additional user identification codes.</td>
</tr>
</tbody>
</table>
Chapter 2: Starting and Exiting BTEQ
Starting a BTEQ Session

<table>
<thead>
<tr>
<th>To access</th>
<th>You need</th>
<th>You may also need</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teradata Database,</td>
<td>a valid Teradata userid and password.</td>
<td>an account identifier (acctid) and/or a Teradata Director Program Identifier (tdpid).</td>
</tr>
</tbody>
</table>

**Obtaining Client System Access**
If you do not have client system access privileges, see your system administrator to obtain a userid, password and any other identifiers needed to log on to the client system.

**Obtaining Teradata Database System Access**
If you do not have Teradata Database access privileges, see your system administrator or Teradata administrator to obtain a userid, password, and any other identifiers needed to access Teradata Database.

**Additional Teradata Database Privileges**
You may require additional Teradata Database privileges to perform the following tasks:

- access certain databases or tables
- create, copy, or modify databases or tables
- create and use Teradata SQL macros
- create and use Teradata stored procedures

Contact your system administrator or Teradata administrator to obtain any additional privileges that you require. For more information on access privileges, refer to *SQL Data Definition Language and Database Design*.

**Starting a BTEQ Session**

This section describes how to start BTEQ on the client system and log on to Teradata Database. Ensure that you have obtained the necessary client and Teradata Database access privileges before continuing with this procedure.

**Starting BTEQ**

To start BTEQ on z/OS, UNIX, or Windows systems, enter BTEQ at the system prompt on your terminal or workstation.

When you invoke BTEQ, the following message displays:

Teradata BTEQ <version> for <OS>. Enter your logon or BTEQ command.

where:

*version* is the release level of your BTEQ software, and

*OS* is the operating system on which you run BTEQ.
Encoding Controls for Unicode Sessions

Workstation BTEQ provides two encoding controls for supporting Unicode. The first control, I/O Encoding, handles the user interface for BTEQ. On input, that includes stdin and RUN files. On output, it includes stdout, stderr, and MESSAGEOUT files.

The second control, Session Character Set Encoding, handles the communication to/from the Teradata Database and the encoding of data files, which includes Import files, Export files, and source files for creating Stored Procedures and User-Defined Functions. Having two encoding values allows one to customize the BTEQ execution environment. All files and communication can be in the same encoding, or UTF8 can be defined for the I/O encoding and UTF16 for the session character set encoding, or visa versa.

The diagram below shows how the two encoding controls affect BTEQ.

Command Line Options

For network-attached systems, BTEQ provides the following command line options:

- `-c [Charset Encoding]`
- `-e [I/O Encoding]`
- `-m`

The `-c` option defines the initial session character set encoding for BTEQ at startup time. This value can be a character set name or a character set code. By default, this option also defines the I/O encoding BTEQ will use if the `-e` and `-m` options are not used (see below).

The `-c` option overrides any charset value defined in the `clispb.dat` file. To start a UTF16 session, this option is required. The SESSION CHARSET command can subsequently be used (certain restrictions apply; see “SESSION CHARSET” on page 276) to change BTEQ’s session.
character set. If the -c option and the clispb.dat file are not used to define a session character set, then BTEQ will default it to ASCII.

The -e option defines the I/O encoding BTEQ will use. This option is only valid when BTEQ is initially set to use a UTF8 or UTF16 session character set (either through the -c option or through the clispb.dat file). The value supplied must be the UTF8 or UTF16 name or code (63 or 62, respectively). Once the -e option has been used, the I/O encoding cannot be changed for that session. It is recommended that the -e option be used before the -c option.

The -m option indicates that I/O encoding for an interactive Unicode session will be encoded in multi-byte characters based on the system locale, instead of true Unicode code points. This option does not take any arguments and requires the use of the -c option, but not the -e option. Without this option, BTEQ will always use the default "C" locale.

**Disclaimer:** When a locale is in effect (activated by the -m option), BTEQ will be limited to the type of characters that it can handle. For example, Thai characters may not be read or displayed when a Russian locale is used. Characters outside the locale will most likely be substituted with a question mark (?)

Example for defining the locale on Windows:
1. Go to CONTROL PANEL.
2. Select REGIONAL AND LANGUAGES OPTION.
3. Select a value for "Language for non-Unicode programs". You may have to install appropriate language support first if the language you require is not available.
4. A reboot may be necessary.

Example for defining the locale on Unix:
1. Use "locale -a" to view the available locales.
2. Set the LC_CTYPE environment variable to an available locale.

When these options and BTEQ commands are specified on the command line, the command line options must be defined first. Examples for using the command line options include:

<table>
<thead>
<tr>
<th>Command Line Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bteq</td>
<td>interactive mode; charset and I/O encoding default to ASCII</td>
</tr>
<tr>
<td>bteq -c ASCII &lt; script</td>
<td>batch mode using an ASCII script; charset and I/O encoding are ASCII</td>
</tr>
<tr>
<td>bteq -c KANJISJIS_0S</td>
<td>interactive mode; charset and I/O encoding are KanjiShift-JIS</td>
</tr>
<tr>
<td>bteq -c UTF8 &lt; script</td>
<td>batch mode using a UTF8 script; charset and I/O encoding are UTF8</td>
</tr>
<tr>
<td>bteq -c UTF16 &lt; script</td>
<td>batch mode using a UTF16 script; charset and I/O encoding are UTF16</td>
</tr>
</tbody>
</table>
Chapter 2: Starting and Exiting BTEQ

LOGON Command

This section describes the LOGON command, which you use to log on to a Teradata Database from BTEQ.

LOGON Command Elements

The table below describes the elements of the LOGON command, some of which may be optional at your installation. For more information on the LOGON command, see “LOGON” on page 206.

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Optional or Required</th>
<th>Specifies</th>
</tr>
</thead>
<tbody>
<tr>
<td>tdpid</td>
<td>Optional</td>
<td>the Teradata Director Program ID of the Teradata server that you are logging on to.</td>
</tr>
<tr>
<td>userid</td>
<td>Required</td>
<td>your user identifier.</td>
</tr>
<tr>
<td>password</td>
<td>Required</td>
<td>the password for your userid.</td>
</tr>
</tbody>
</table>
Chapter 2: Starting and Exiting BTEQ
Logging on to Teradata Database

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Optional or Required</th>
<th>Specifies</th>
</tr>
</thead>
<tbody>
<tr>
<td>acctid</td>
<td>Optional</td>
<td>the account identifier for your userid.</td>
</tr>
</tbody>
</table>

**Specifying the tdpid**

The tdpid is the Teradata Director Program ID associated with a particular Teradata server. Every Teradata server is assigned a unique tdpid. If you do not specify a tdpid and are logging on for the first time, BTEQ logs on to the default tdpid. If you have already logged on, BTEQ logs on to the current tdpid.

**Specifying Your userid**

Your userid is an assigned character string that identifies an individual user, and is associated with a specific password. A userid can also be associated with a specific account identifier, or acctid. Enter your userid followed by a semicolon, a comma, or a blank character.

**Specifying Your password**

Your password is an assigned character string that is associated with your userid. You can enter your password using either uppercase or lowercase letters. You cannot include the semicolon (;) character within or at the end of your password unless the entire password string is enclosed in quotes.

**Specifying Your acctid**

Your acctid is an optional character string that identifies your individual user account, and is associated with a specific userid. Enter your acctid using apostrophes (single quotes) as delimiters immediately before and after the character string. If your account identifier contains an apostrophe, use full double quotes as delimiters, or repeat the apostrophe character in the account identifier.

**Logging on to Teradata Database**

This section contains the procedures for logging on to Teradata Database after invoking BTEQ on the client system.
Log on to Teradata Database

Logging On in Interactive Mode

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Result or Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Enter the LOGON command as follows: \LOGON tdpid/userid,, 'acctid'</td>
<td>BTEQ clears the input area and prompts you for your password. Note: Omit acctid if you have not been issued one. If you omit userid, BTEQ prompts you to enter it.</td>
</tr>
<tr>
<td>2</td>
<td>Enter your Teradata Database password: *******</td>
<td>BTEQ does not display the password. If the logon is successful, BTEQ displays the successful logon message.</td>
</tr>
</tbody>
</table>

Logging On in Batch Mode

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Result or Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Submit the LOGON command in an input file, including the password, as follows: \LOGON tdpid/userid, password, 'acctid'</td>
<td>Note: Omit acctid if you have not been issued one.</td>
</tr>
</tbody>
</table>

Bypassing Logon Prompts

If you are using BTEQ on a network-attached system, you can log on to Teradata Database without the system prompting you for your DBS username and password by setting the LOGONPROMPT command to OFF (refer to “LOGONPROMPT” on page 211 for more information).

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Result or Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Enter the LOGONPROMPT command: \SET LOGONPROMPT OFF</td>
<td>The system will not prompt you to provide or qualify userid and password information.</td>
</tr>
<tr>
<td>2</td>
<td>Enter the LOGON command: \LOGON tdpid/</td>
<td>If the logon is successful, BTEQ displays the successful logon message.</td>
</tr>
</tbody>
</table>

Successful Logon Message

The successful logon message displays the version of the Teradata Database that you have logged on to and the transaction semantics and session charset in effect for the current session, in a format similar to the following:

*** Logon successfully completed.
*** Teradata Database Release is 13.10
*** Teradata Database Version is 13.10
*** Transaction Semantics are BTET
*** Character Set Name is 'ASCII'
Failed Logon Message

If the logon is not successful, BTEQ displays the failed logon message and prompts you for another logon:

*** Error: Logon failed!
*** Total elapsed time was n second(s).
Teradata BTEQ for (system). Enter your logon or BTEQ command:

Retrying a Failed Logon Attempt

If your logon failed, repeat the logon procedure, being especially careful when you enter your password, which is not displayed on the screen. If repeated attempts are unsuccessful, contact your system administrator.

Logging Off Teradata Database / Exiting Database

There are three termination commands that you can use to log off from Teradata Database and exit BTEQ:

<table>
<thead>
<tr>
<th>If you want to...</th>
<th>Use this command...</th>
</tr>
</thead>
<tbody>
<tr>
<td>end the current Teradata Database sessions without exiting BTEQ,</td>
<td>LOGOFF</td>
</tr>
<tr>
<td>end the current Teradata Database sessions and exit BTEQ,</td>
<td>EXIT or QUIT</td>
</tr>
<tr>
<td>exit BTEQ when you are not logged on to Teradata Database,</td>
<td>EXIT or QUIT</td>
</tr>
</tbody>
</table>

You can enter the LOGOFF, EXIT, and QUIT commands at your terminal or workstation only in response to the BTEQ command prompt. Any previous command or request must be completed. To interrupt or abort BTEQ before the previous command or request completes, use the Break key as described in “Interrupting and Aborting Requests” on page 81.

LOGOFF Command

The LOGOFF command ends the current Teradata Database sessions without exiting BTEQ. If you execute a LOGOFF command while a transaction is in progress, BTEQ aborts the transaction and backs out any changes that had been made to the database.

Note: Executing a LOGOFF command does not reset the format settings to their default values. The next sessions that you log on to within the current set will inherit the previous format settings. To reset the format options to their default values, enter the DEFAULTS command before your next LOGON command.

LOGOFF Example

Enter the LOGOFF command at the BTEQ command prompt:
Chapter 2: Starting and Exiting BTEQ

Changing Your Password

Depending on the parameters set by your Teradata administrator, your Teradata Database password may expire periodically. BTEQ requires that you change your expired password before logging on to Teradata Database.

When Your Password Expires

When your Teradata Database password expires, BTEQ displays the following message and prompts you to enter a new password:

```plaintext
*** Warning 3032 USER PASSWORD HAS EXPIRED.
Enter new password:
```
Replacing Your Expired Password

Follow this procedure to replace your expired password. You can enter up to 30 characters using uppercase and lowercase letters. The password cannot include the semicolon character.

1. Enter your new password at the prompt, for example:
   
   Enter new password: password
   
   Note: The system will not display your password as you enter it.

2. Reenter your password to validate it.
   
   If both entries are identical, and the logon is successful, BTEQ displays:
   
   * * * Logon successfully completed.
   * * * Total elapsed time was 1 minutes and 13 seconds
   
   If both entries are not identical, BTEQ displays the following message and repeats the prompt for a new password:
   
   Password not matched, please try again!
   
   Enter new password

Replacing Your Password at Any Time

For information on changing your password at any time, refer to the MODIFY USER statement examples in SQL Data Definition Language.

Specifying a Different Teradata Server

When you start a BTEQ session, the Teradata server that you access is specified by the tdpid. You can specify a different server for a single logon, or for the current BTEQ session and all subsequent logons within that session.

System Default tdpid

Channel-attached system configurations include a default tdpid. (If you do not know the default tdpid for your system, check with your system administrator.) You cannot change the default tdpid for your system.

Changing the tdpid

You can change the tdpid using one of the following methods:

<table>
<thead>
<tr>
<th>If you want to...</th>
<th>Then use...</th>
</tr>
</thead>
<tbody>
<tr>
<td>change the server for a single logon,</td>
<td>the tdpid option of the LOGON command.</td>
</tr>
<tr>
<td>change the server for all logons within the current session,</td>
<td>the TDP command.</td>
</tr>
</tbody>
</table>
Regardless of whether you use the TDP command or the LOGON command to specify a different Teradata Database, the newer \textit{tdpid} replaces the older \textit{tdpid} as the default for the current BTEQ session. For each new session, BTEQ begins with the standard default \textit{tdpid} for your client.

**Changing the Server for a Single Logon**

Use the \textit{tdpid} option of the LOGON command to specify a different Teradata server for a single logon:

```
.LOGON tdpid/username, password
```

\textbf{Note:} You can enter the TDP commands whenever you want to specify a different Teradata Database for subsequent logons during the current session.

**Changing the Server for the Current Session**

Use the BTEQ TDP command to specify a different Teradata server for all logons within the current session:

```
.SET TDP tdpid
```

Entering the TDP command specifies the Teradata Database that BTEQ accesses for all the logons during the current BTEQ session.

**Specifying Multiple Sessions**

BTEQ can ask for dedicated Teradata sessions in sets, so that multiple SQL requests can run in parallel within a given set. This is especially helpful when you need to process a high volume of repetitive tasks whose order does not matter, such as loading a large number of rows onto a database. In this case, having a number of parallel sessions with each handling some of the requests can substantially speed up operations.

**Maximum Number of Sessions in a Set**

Though BTEQ can support a maximum of up to 200 sessions, the actual maximum depends on the configuration of your system and your terminal or workstation. The maximum for your system could be less than 200, and as few as 16 for PC workstations.

**Setting the Number of Sessions**

BTEQ supports a maximum of 200 sessions. Specify the number of sessions in a set by entering the SESSIONS command prior to the LOGON command. The default number of sessions is 1. (See “SESSIONS” on page 287 for more details.)

To specify more than one session in a set:

✔ At the BTEQ prompt:

```
Teradata BTEQ <version> for <OS>. Enter your logon or BTEQ command.
```
enter:

```
.SET SESSIONS n
```

where \( n \) is the number of sessions.

## Getting Help

You can display online help for BTEQ and its commands by using the HELP BTEQ command at the command prompt.

The BTEQ help display provides a brief description of BTEQ, a listing of each command in alphabetical order, and additional notes on input. The command listings include syntax descriptions in text format rather than graphical format.

## Syntax Conventions

The online help represents command syntax using the following conventions:

<table>
<thead>
<tr>
<th>This character...</th>
<th>Indicate(s)...</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ ] (Brackets)</td>
<td>optional choices. Choices are enclosed in brackets. You can specify one or none of the options.</td>
</tr>
<tr>
<td>{ } (Braces)</td>
<td>required choices. Choices are enclosed in brackets. You must specify one of the options.</td>
</tr>
<tr>
<td></td>
<td>(vertical bar)</td>
</tr>
<tr>
<td>UPPERCASE characters</td>
<td>reserved words. Words are shown in uppercase. You must enter reserved words exactly as they are shown.</td>
</tr>
<tr>
<td>lowercase characters</td>
<td>variables. You must supply values for the variables.</td>
</tr>
</tbody>
</table>

## Notes on Command Input

The online help includes the following information:

- All BTEQ commands must be preceded by a dot (.) character.
- BTEQ commands may end with a semicolon (;).
- Any command that does not start with a dot is assumed to be a Teradata SQL query, and will be submitted to Teradata Database.
- BTEQ commands may be executed in a Teradata SQL macro by using the ECHO statement, for example:

  ```
  ECHO '.HELP BTEQ';
  ```
CHAPTER 3
Using BTEQ

This chapter describes the fundamentals of using BTEQ, including:

- Using Teradata SQL in BTEQ
- Using Comments in a BTEQ Session
- UNIX Signals
- Using Scripts, Run Files, Macros, and Stored Procedures
- Testing and Branching
- Using the REPEAT Command
- Interrupting and Aborting Requests
- Entering Operating System Commands from a BTEQ Session
- Handling Errors
- I/O Errors and Abends
- BTEQ Large File (LF) Support (for files >2GB)

Using Teradata SQL in BTEQ

All database requests are expressed in Teradata Structured Query Language (Teradata SQL). A Teradata SQL request contains one or more Teradata SQL statements. You can use Teradata SQL statements to:

- Define data - create and modify data structures.
- Select data - query a database.
- Manipulate data - insert, delete, and update data.
- Create Teradata SQL macros - store and execute sequences of Teradata SQL statements as a single operation.
- Control data - define databases and users, establish privileges, and secure data.

BTEQ does not support embedded control characters within quoted strings for SQL requests. Examples of unsupported control characters include a null, carriage return, or linefeed character.

Request Types

There are two types of Teradata SQL requests. A single-statement request is a single Teradata SQL statement sent as a request. A multi-statement request is two or more statements that are sent as a request.
Semicolon placement in relation to the rest of a line (for example, at the beginning or end) determines whether a statement is processed as a single-statement request or a multi-statement request.

<table>
<thead>
<tr>
<th>To submit...</th>
<th>Place a semicolon...</th>
</tr>
</thead>
<tbody>
<tr>
<td>a single-statement request</td>
<td>at the end of the statement as the last nonblank character of the line.</td>
</tr>
<tr>
<td>a multi-statement request</td>
<td>at the beginning of the line that begins the second through the final statement, and at the end of the final statement as the last nonblank character of the line.</td>
</tr>
</tbody>
</table>

**Single-Statement Example**

BTEQ submits the following statements to Teradata Database as three single-statement requests:

```sql
SELECT * FROM Employee;
DELETE FROM Employee WHERE Name = 'Inglis C' AND Empno = 10014;
SELECT Name FROM Employee;
```

**Multi-Statement Example**

To submit the same three statements as a multi-statement request, enter:

```sql
SELECT * FROM Employee
; DELETE FROM Employee WHERE Name = 'Inglis C' AND Empno = 10014
; SELECT Name FROM Employee;
```

BTEQ sends only one request at a time to any one available Teradata Database session. If an error occurs that indicates the request failed but can be retried, BTEQ attempts to resubmit the request. However, this is the extent to which BTEQ takes action on a retryable error.

Because BTEQ passes SQL to the database without interpreting it, it is not transaction state aware. It exercises no control over a transaction’s state, so the state must be driven solely by the method that is used to include BT/ET/ABORT (Teradata mode) or COMMIT/ROLLBACK (ANSI mode). BTEQ’s unawareness, in addition to the differences between ANSI and Teradata transaction semantics modes, needs to be taken into consideration when it is important for an entire transaction associated with one or more submitted requests to be entirely retryable, or when a transaction must span requests.

In Teradata semantics mode, regardless of the method used to identify its transaction (implicit, as in a multi-statement request or macro execution, or explicit, via BT/ET), a request or statement’s failure results in rollback of that transaction. It is the user’s responsibility to decide what to do as a result of such rollbacks: redrive the rolled-back portion of the transaction (presumably after correcting whatever condition led to the error), abandon the transaction (that is, bypass the remaining requests/statements that would have been submitted as part of the transaction), or do something else appropriate for the particular circumstances.
In ANSI semantics mode, you are always in a transaction unless you have just completed or reversed one by issuing COMMIT or ROLLBACK, respectively. An ANSI mode statement failure results in rollback of the request, not the transaction. It is again the user’s responsibility to correct and redrive the request, submit a ROLLBACK statement to rollback the remainder of the already-processed transaction, or do something else appropriate for the particular circumstances.

For a complete discussion of Teradata vs. ANSI modes and their effect on transaction semantics, see *SQL Request and Transaction Processing*.

**Multi-Statement Processing**

In a multi-statement request, the semicolons that begin the line are used only to separate the statements. BTEQ does not submit any of the statements to Teradata Database until it encounters a semicolon as the last nonblank character of a line. At that time, BTEQ sends all of the statements to Teradata Database for processing as one single request.

**Note:** You cannot use an ANSI comment statement in conjunction with a semicolon character to indicate the final statement of a multi-statement request. If the semicolon appears after an ANSI comment, it will be treated as part of the comment text string. If it appears before an ANSI comment, it will not qualify as the last nonblank character on the line.

For complete documentation on Teradata SQL, refer to *SQL Fundamentals*.

**Negating Requests**

At times, it is necessary to negate a statement(s) you have entered because the statement would have an undesired effect on table data. If you are using BTEQ interactively to enter either a single-statement or multi-statement request, and you have already pressed ENTER to continue typing on the next line of the request, you cannot use the backspace function to delete characters on the prior line. Instead, you can:

1. Continue the request with an additional statement that has invalid syntax.
2. Press ENTER.

This causes all statements to be ignored. (This assumes you have not yet pressed ENTER after the final trailing semicolon, an action that sends the request to the database.)

**Example**

If you have typed the following, but have not pressed ENTER after the final trailing semicolon:

```sql
SELECT DATE
;
```

you could continue typing to add an additional, invalid statement. For example:

```sql
SELECT DATE
; junk
```

After you pressed ENTER, an error message similar to the following would be returned:

```plaintext
*** Failure 3706 Syntax error: expected something between ';' and the word 'junk'.
   Statement# 3, Info =20
```
Continuing SQL

A Teradata SQL request is automatically continued from one line to the next unless the line ends in a semicolon, a left or right parenthesis, or a single-quote or double-quote character. However, the following conditions apply:

<table>
<thead>
<tr>
<th>The SQL will not terminate if...</th>
<th>For example...</th>
</tr>
</thead>
<tbody>
<tr>
<td>termination characters are embedded within a comment,</td>
<td>exec /* exec macro current; */ macro xyz;</td>
</tr>
<tr>
<td>a semicolon is embedded in an unsatisfied parenthesis, or a single- or double-quote pair,</td>
<td>create macro xyz as (select * from dbc.tables; );</td>
</tr>
<tr>
<td>the right parenthesis, single- and double-quote characters are the first of a pair whose companion has not yet appeared,</td>
<td>select * from dbc.tables where databasename='alctest';</td>
</tr>
</tbody>
</table>

Using Comments in a BTEQ Session

You can use two types of comments in a BTEQ session:

- BTEQ comments
- ANSI comments

BTEQ Comment Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Function</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>* (asterisk)</td>
<td>Begins a BTEQ comment</td>
<td>It must be the first nonblank character on the line.</td>
</tr>
<tr>
<td>Text string</td>
<td>Contains the comment text</td>
<td>Can be as many characters as fills the line, less one for the terminator character.</td>
</tr>
<tr>
<td>&lt;cr&gt; (carriage return)</td>
<td>Terminates a BTEQ comment</td>
<td>The rest of the line, after the ending character, must be blank.</td>
</tr>
<tr>
<td>&lt;lf&gt; (line feed)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>* asterisk</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

BTEQ Comment Example

BTEQ comments begin with an asterisk (*), contain text, and end with one of three termination characters, as in the following example:

* Comment text*
Continuing BTEQ Comments
To continue a comment to the next line, create two consecutive comments, as shown in the following example:

* This table defines all parameters <cr>
* in the Department table <cr>
* which is normally updated on a weekly basis <cr>

ANSI Comment Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Function</th>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>-- (double dash)</td>
<td>Begins an ANSI comment</td>
<td>It must be the first nonblank character on the line.</td>
</tr>
<tr>
<td>Text string</td>
<td>Contains the comment text</td>
<td>Can be as many characters as fills the line, less one for the terminator character.</td>
</tr>
<tr>
<td>&lt;cr&gt; (carriage return)</td>
<td>Terminates an ANSI comment</td>
<td>The rest of the line, after the ending character, must be blank.</td>
</tr>
<tr>
<td>&lt;lf&gt; (line feed)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ANSI Comment Example
ANSI comments begin with a double-dash symbol (--), contain text, and end within a single line with either a carriage return <cr> or a line feed <lf>, as in the following example:

-- ANSI comment example text<cr>

Mixing Comments
Do not mix any of the following:
- BTEQ and ANSI comments
- BTEQ comments and Teradata SQL comments
- ANSI comments and Teradata SQL comments

Concatenating BTEQ Comments
Do not use a BTEQ comment after:
- Teradata SQL commands
- BTEQ commands
- A semicolon terminator character

Illegal Comments
You cannot use:
- BTEQ or ANSI comments in a Teradata SQL macro. (Use REMARK as described in “REMARK” on page 249.)
Chapter 3: Using BTEQ

Entering IPv4 and IPv6 Network Addresses

Network attached clients support both IPv4 and IPv6 network addressing. There are three ways to provide IP addresses to BTEQ.

- via the `SET TDP` command
- via the `LOGON` command
- via the `dbcname` parameter in `clispb.dat`

An IPv6 address must be enclosed in square brackets `[ ]`. If a port number is also provided, place it outside the brackets and separate it from the IP address with a colon. The following are valid IPv4/IPv6 address formats.

- `<IPv4 address>`
- `<IPv4 address>:`<port>
- `[<IPv6 address>]`
- `[<IPv6 address>]:<port>`

UNIX Signals

Signals are predefined messages sent between two UNIX processes to communicate the occurrence of unexpected external events, or exceptions. You cannot incorporate the UNIX signals that BTEQ uses into an access module or routine for BTEQ. Doing so will cause an error.

Running BTEQ Batch Jobs

This section describes the process of running batch jobs to submit BTEQ commands and Teradata SQL.

Running Under z/OS

You can submit BTEQ commands and Teradata SQL statements using z/OS in either of two ways:

- within a previously created input file
- as part of the Job Control Language (JCL) program input stream

Sample Procedure

To execute BTEQ z/OS, use the BTEQ JCL procedure TDSBTEQ located in TERADATA.PROCLIB.
**Job Card Example**

You can also use a standard z/OS JCL card to submit the BTEQ job to Teradata Database, as in the following example:

```
//YNBTQ  Job 1,'Your Name',CLASS=B,Notify=YN,
     MSGCLASS=A,MSGLEVEL=(1,1)
//BTEQ  EXEC PGM=BTQMAIN
//STEPLIB DD DSN=TERADATA.APPLOAD,DISP=SHR
    DD DSN=TERADATA.TRLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*,DCB=(LRECL=137)
//SYSABEND DD SYSOUT=*    DD SYSOUT=*  
//SYSTERM DD SYSOUT=*       DD DATA,DLM=#
```

<table>
<thead>
<tr>
<th>This statement...</th>
<th>Identifies or defines...</th>
</tr>
</thead>
<tbody>
<tr>
<td>//YNBTQ  Job 1,'Your Name',CLASS=B,Notify=YN, MSGCLASS=A,MSGLEVEL=(1,1)</td>
<td>the job.</td>
</tr>
<tr>
<td>//BTEQ  EXEC PGM=BTQMAIN</td>
<td>and executes the BTEQ program.</td>
</tr>
<tr>
<td>//STEPLIB DD DSN=TERADATA.APPLOAD,DISP=SHR</td>
<td>the client library where BTEQ resides.</td>
</tr>
<tr>
<td>//    DD DSN=TERADATA.TRLOAD,DISP=SHR</td>
<td></td>
</tr>
<tr>
<td>//SYSPRINT DD SYSOUT=*,DCB=(LRECL=137)</td>
<td>defines the output file and executes printing.</td>
</tr>
<tr>
<td>//SYSABEND DD SYSOUT=*</td>
<td>the dump file in case the channel-attached system fails. (Always include a SYSABEND or SYSDUMP card to capture a dump if the system fails.)</td>
</tr>
<tr>
<td>//SYSTERM DD SYSOUT=*</td>
<td>the standard error file for error messages redirected by the BTEQ ERROROUT command.</td>
</tr>
<tr>
<td>//SY SIN DD DATA,DLM=#</td>
<td>precedes the BTEQ script.</td>
</tr>
</tbody>
</table>

**Note:** Use DD DATA,DLM=# to indicate the end of the file instead of //SYSIN DD*. The * (asterisk) character is interpreted as the Teradata SQL comment indicator.

**Running Other Client Systems**

When running UNIX or Windows, you can provide BTEQ commands and Teradata SQL statements in a file that you have created and stored using your workstation editor. In this case, always use an editor that does not include line numbers in its output file. Because BTEQ uses all of the information on each line, line numbers introduced by your editor are interpreted as part of your input stream.
Switching from Interactive to Batch Mode

When switching a session from interactive to batch mode, ensure that the script to be run is in the same directory as BTEQ. First log on, then run the file(s). For example:

`.run file bteq00.inv`  
`.run file bteq01.inv`  
`etc.`

Defining Input and Output Files for BTEQ in Batch Mode

Input files (or input stream files) are files that you identify as sources of BTEQ commands, SQL requests, or data that you want to write to Teradata Database. Similarly, output files are files that you identify as destinations for data or error messages retrieved from Teradata Database.

This section explains how to define the input and output files required to run BTEQ in batch mode. It also describes how to define additional files required by specific BTEQ commands in both batch and interactive modes.

File Types

BTEQ uses three types of input files and three types of output files:

<table>
<thead>
<tr>
<th>Input Files</th>
<th>Output Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Input</td>
<td>Standard Output</td>
</tr>
<tr>
<td>Import File</td>
<td>Export File</td>
</tr>
<tr>
<td>Run File</td>
<td>Standard Error Output</td>
</tr>
</tbody>
</table>

Use the appropriate BTEQ commands or file specifications, as described in the following subsections, to specify your input and output files.

Defining Input and Output Files

You must define an input file and an output file to use BTEQ in batch mode. In interactive mode, BTEQ defines these files for you automatically when you log on.

<table>
<thead>
<tr>
<th>File Type</th>
<th>ddname</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>SYSIN</td>
<td>contains the Teradata SQL statements and BTEQ commands that make up a BTEQ job.</td>
</tr>
<tr>
<td>Output</td>
<td>SYSPRINT</td>
<td>receives the results of your job from Teradata Database.</td>
</tr>
</tbody>
</table>
To execute BTEQ in interactive mode, either clear the SYSIN file or define it as TERMINAL.

**Note:** BTEQ writes lowercase letters to its output file. If your printer does not support lowercase letters, use FOLDMODE on printer output to map lowercase characters to uppercase characters.

### Defining Additional Files

You must also define files for use with the following commands, for both batch and interactive operations.

<table>
<thead>
<tr>
<th>Command Name</th>
<th>File Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPILE</td>
<td>input</td>
</tr>
<tr>
<td>ERROROUT</td>
<td>output</td>
</tr>
<tr>
<td>EXPORT</td>
<td>output</td>
</tr>
<tr>
<td>IMPORT</td>
<td>input</td>
</tr>
<tr>
<td>RUN</td>
<td>input</td>
</tr>
</tbody>
</table>

### Specifying a Standard Input File

A standard input file is an input file that you have defined as the SYSIN file. It executes automatically in batch mode when you invoke BTEQ. If you do not specify a standard input file, then you must provide the input stream interactively through your terminal or workstation keyboard.

Depending on the configuration of your system, you can specify a standard input file either when you invoke BTEQ, or before you log on to Teradata Database.

To specify an input file:

<table>
<thead>
<tr>
<th>If your system is...</th>
<th>Then use...</th>
</tr>
</thead>
<tbody>
<tr>
<td>channel-attached</td>
<td>either the BTEQ TDSBTEQ JCL procedure or a DD statement in a standard JCL file to specify your input file as the SYSIN file before invoking BTEQ.</td>
</tr>
<tr>
<td>network-attacheda</td>
<td>the &lt; character to specify your input file when you invoke BTEQ.</td>
</tr>
</tbody>
</table>

a. The use of the -c and -e command line options or the “charset_id” value in the clispb.dat file is highly recommended when a Unicode input file is used. Otherwise, BTEQ will try to interpret the input file as Unicode and set the session character set accordingly, as follows. BTEQ will check the input file for a UTF16 or UTF8 BOM and will automatically change the session character set to the appropriate Unicode encoding. The endianness of a UTF16 BOM must match that of the machine BTEQ is running on. If a BOM-less UTF16 input file is used, BTEQ will validate the first character and will automatically change the session character set to UTF16. Note that BTEQ will not automatically change the session character set for a BOM-less UTF8 input file. If none of the above apply, BTEQ will assume the input file is not Unicode and will default the session character set to ASCII.
For more information and examples of specifying a standard input file, refer to “Using Scripts, Run Files, Macros, and Stored Procedures” on page 61.

**Using a Here Document for Standard Input Files**

In scenarios where environment variables need to be used as part of the BTEQ script, a Here document, which allows you to specify a string literal in a command-line shell, is a possible option.

The following example outlines a UNIX script as a Here document.

```
#!/bin/sh
#########################################
bteq <<EOI
export DATADIR=/datasvc/data
.LOGON mydbs/myid,mypw;
.EXPORT REPORT FILE=$DATADIR/output
SELECT * from table_name;
.EXPORT RESET
.LOGOFF
.EXIT
EOI
#########################################
```

**Specifying a Standard Output File**

A standard output file is an output file that you have defined as the SYSPRINT file. It executes automatically in batch mode when you invoke BTEQ. If you do not specify a standard output file, or if you define your terminal or workstation as the SYSPRINT file, then BTEQ displays all of the output reports and messages on your terminal or workstation screen.

Depending on the configuration of your system, you can specify a standard output file either when you invoke BTEQ, or before you log on to Teradata Database.

To specify an output file:

<table>
<thead>
<tr>
<th>If your system is...</th>
<th>Then use...</th>
</tr>
</thead>
<tbody>
<tr>
<td>channel-attached</td>
<td>either the BTEQ TDSBTEQ JCL procedure or a DD statement in a standard JCL file to specify your output file as the SYSPRINT file before invoking BTEQ.</td>
</tr>
<tr>
<td>network-attached</td>
<td>the &gt; or &gt;&gt; character to specify your output file when you invoke BTEQ.</td>
</tr>
</tbody>
</table>

**Using the > or >> Characters**

Use the > and >> characters to specify output files as follows:

<table>
<thead>
<tr>
<th>If you specify a filename using...</th>
<th>And the filename...</th>
<th>BTEQ will...</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; (single character)</td>
<td>is new</td>
<td>create the new file as the standard output file.</td>
</tr>
</tbody>
</table>
Defining Input and Output Files for BTEQ in Batch Mode

Chapter 3: Using BTEQ

For more information and examples of specifying a standard output file, refer to “Using Scripts, Run Files, Macros, and Stored Procedures” on page 61.

**Specifying Output File Block Size**

When you define output files, specify a BLKSIZE that is large enough to accommodate the results that you expect.

<table>
<thead>
<tr>
<th>For this type of job...</th>
<th>Specify a region of...</th>
</tr>
</thead>
<tbody>
<tr>
<td>single interactive mode session</td>
<td>1024K</td>
</tr>
<tr>
<td>single-session batch job</td>
<td>1024K</td>
</tr>
<tr>
<td>large, multi-session job</td>
<td>4096K</td>
</tr>
</tbody>
</table>

**Specifying Output File Logical Record Length**

Your LRECL specification for output files must indicate the appropriate record length, plus five additional bytes:

- One byte for carriage control character
- Four bytes to accommodate variable format

*Note:* Carriage control for EXPORT REPORT/DIF files is not supported for Unicode when using z/OS BTEQ.

**Specifying an Import File**

An import file is an input file that you use primarily for writing new data to Teradata Database. BTEQ transfers the import file data to Teradata Database in response to the USING modifier in a Teradata SQL request.

To specify an import file:

<table>
<thead>
<tr>
<th>If your system is...</th>
<th>Then...</th>
</tr>
</thead>
<tbody>
<tr>
<td>channel-attached</td>
<td>use a z/OS DD statement to define an import file, then execute a BTEQ IMPORT command to open the file before entering your SQL request.</td>
</tr>
<tr>
<td>network-attached</td>
<td>you do not have to define import files, but you do have to use the IMPORT command to open them before making your SQL request.</td>
</tr>
</tbody>
</table>
Specifying Files for User-Defined Functions

When you cannot accomplish a programming task with SQL’s built-in functions, you can write a user-defined function in standard C, C++, or Java programming languages to extend your set of available functions.

The C, C++, or Java files can reside on the server or the client. The server will call for the transfer of client-resident files as needed to satisfy input requirements for the CREATE/REPLACE FUNCTION request. Source files must be encoded as ASCII (workstation) or EBCDIC (mainframe), no matter what the current session character set is. Note, the same transfer protocol is employed when using client-resident files to create User-Defined Methods and External Stored Procedures.

**Note:** Starting with DBS 13.10 and BTEQ 13.10, the encoding expected for file name values is based on session character set rather than solely on the platform-appropriate single-byte character sets of ASCII or EBCDIC. This means that, for Unicode sessions only, the transfer protocol will not be forward- or backward-compatible when a 13.10 version of either DBS or BTEQ is used with an older version of the other.

**Workstation BTEQ**

If workstation BTEQ cannot locate a file based on the name requested by the database, it attempts to append an extension to the name, then tries again. The extension is derived from information from Teradata Database about the associated source language and file type. Workstation BTEQ derives extension for the following file types:

- .c
- .cpp
- .h
- .o (when using UNIX BTEQ)
- .obj (when using Windows BTEQ)

**Mainframe BTEQ**

Mainframe BTEQ only accepts DDNAMES in the EXTERNAL clause of the CREATE FUNCTION statement. Therefore, it makes no attempt to derive file types for file names.

**Transferring Source or Include Files Using Mainframe BTEQ**

When transferring source or include files using mainframe BTEQ, the record gaps that delineate end-of-line are replaced by BTEQ with carriage returns so that they can be mapped to meaningful code points by the database. You do not need to do anything to the file. This is the only file manipulation and files of other types are transferred without modification.

The full specification of the syntax, format, and rules for both creating and invoking UDFs is beyond the scope of this document. See *SQL External Routine Programming* for details on how to create and invoke user-defined functions.
Specifying an Export File

An export file is an output file that you use primarily for storing data read from Teradata Database in response to a subsequent SQL request.

To specify an export file:

<table>
<thead>
<tr>
<th>If your system is…</th>
<th>Then…</th>
</tr>
</thead>
<tbody>
<tr>
<td>channel-attached</td>
<td>use a z/OS DD statement to define an import file, then execute a BTEQ EXPORT command to open the file before entering your SQL request.</td>
</tr>
<tr>
<td>network-attached</td>
<td>you do not have to define import files, but you do have to use the EXPORT command to open them before making your SQL request.</td>
</tr>
</tbody>
</table>

If you do not define an export file name, the default file name for the exported data is FILE ddname.

**Note**: SYSPRINT is not a valid ddname for the BTEQ EXPORT command. You cannot export to the SYSPRINT file.

Specifying a Run File

A run file is an input stream file that has not been defined as the SYSIN file and, consequently, does not execute automatically when you invoke BTEQ. Instead, you must use the BTEQ RUN command to execute a run file.

To specify a run file:

<table>
<thead>
<tr>
<th>If your system is…</th>
<th>Then…</th>
</tr>
</thead>
<tbody>
<tr>
<td>channel-attached</td>
<td>use a z/OS DD statement to define an input file as a run file before executing a BTEQ RUN command.</td>
</tr>
<tr>
<td>network-attached</td>
<td>you do not have to define run files.</td>
</tr>
</tbody>
</table>

For more information and examples of defining run files, refer to “Using Scripts, Run Files, Macros, and Stored Procedures” on page 61.

Specifying a Standard Error Output File

By default, BTEQ writes error messages to the standard output file (STDOUT), along with the data retrieved from Teradata Database in response to your BTEQ commands and SQL requests. On channel-attached systems, and some network-attached systems, you can specify a standard error output file (STDERR) for error messages.

For more information about specifying a standard error output file, refer to “Handling Errors” on page 85 and to “ERROROUT” on page 149.

Depending on the configuration of your system, you can specify a standard error output file either when you invoke BTEQ or before you log on to Teradata Database.
To specify a standard error output file:

<table>
<thead>
<tr>
<th>If your system is...</th>
<th>Then...</th>
</tr>
</thead>
<tbody>
<tr>
<td>channel-attached</td>
<td>use either the BTEQ TDSBTEQ JCL procedure or a DD statement in a standard JCL file to define the ddname of SYSTERM for the STDERR device before invoking BTEQ.</td>
</tr>
<tr>
<td>channel-attached</td>
<td>you can also use the BTEQ TSO command to define the ddname of SYSTERM for the STDERR device for channel-attached systems.</td>
</tr>
<tr>
<td>network-attached</td>
<td>use the &gt; or &gt;&gt; characters to define a standard error output file when you invoke BTEQ.</td>
</tr>
</tbody>
</table>

### Using the > or >> Characters

Use the > and >> characters to specify output files in a UNIX environment, as follows:

<table>
<thead>
<tr>
<th>If you specify a filename using...</th>
<th>And the filename...</th>
<th>BTEQ will...</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; (single character)</td>
<td>is new</td>
<td>create the new file as the standard error output file.</td>
</tr>
<tr>
<td>&gt; (single character)</td>
<td>already exists</td>
<td>overwrite the existing file with new records.</td>
</tr>
<tr>
<td>&gt;&gt; (double characters)</td>
<td>is new</td>
<td>create the new file as the standard error output file.</td>
</tr>
<tr>
<td>&gt;&gt; (double characters)</td>
<td>already exists</td>
<td>add the new records to the end of the existing file.</td>
</tr>
</tbody>
</table>

**Note:** You can also use the >& and >&1 characters to specify the name of a new file to be created for both standard output and error messages.

### Shell Output Stream Differences on SUN Systems

On SUN systems, the direction of the standard output stream and error messages is not the same under the sh shell and the C shell.

<table>
<thead>
<tr>
<th>If you are running...</th>
<th>And you enter...</th>
<th>Then BTEQ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>the sh shell</td>
<td>&gt;out</td>
<td>sends error messages to the file called out.</td>
</tr>
<tr>
<td>the sh shell</td>
<td>&gt;out&amp;</td>
<td>sends the standard output stream to the system console and runs BTEQ in the background.</td>
</tr>
<tr>
<td>the sh shell</td>
<td>&gt;out&gt;&amp;1</td>
<td>sends both error messages and the standard output stream to the file out.</td>
</tr>
<tr>
<td>the C shell</td>
<td>&gt;out</td>
<td>sends the standard output stream to the file out, and error messages to your workstation screen.</td>
</tr>
</tbody>
</table>
Specifying Error Output Files on Windows

When specifying a standard error output file on Windows,

- `>out` sends standard output to files, “OUT”.
- error messages cannot be redirected.

After you have specified a standard error output file, use the BTEQ ERROROUT command to direct error messages to it.

Using Scripts, Run Files, Macros, and Stored Procedures

Depending on where you store them and how you execute them, files that contain your BTEQ commands and Teradata SQL statements are called scripts, run files, macros, or stored procedures:

<table>
<thead>
<tr>
<th>This file type...</th>
<th>Which is...</th>
<th>Is appropriate for...</th>
</tr>
</thead>
<tbody>
<tr>
<td>script, or input stream file</td>
<td>a disk file contained within the SYSIN file on your channel-attached system</td>
<td>a set of commands and requests that produces a standard report or display, and that has been thoroughly tested.</td>
</tr>
<tr>
<td>run file</td>
<td>a disk file that is not contained within the SYSIN file</td>
<td>a set of commands and requests that you need to use over and over again, for example, to produce the same report or display on a daily or weekly basis.</td>
</tr>
<tr>
<td>stored procedure</td>
<td>a file that is created, compiled and stored on Teradata Database, and is executed in response to a Teradata SQL CALL statement</td>
<td>a number of users who need to perform the same operations on a regular basis.</td>
</tr>
<tr>
<td>macro</td>
<td>a file that is created and stored on Teradata Database, and is executed in response to a Teradata SQL EXECUTE statement</td>
<td>a number of users who need to produce the same report or display on a regular basis.</td>
</tr>
</tbody>
</table>

Use the text editor on your system to create and maintain BTEQ scripts, stored procedure source text, and run files. Use the Teradata SQL CREATE MACRO and ECHO statements to create BTEQ macros.

Use the COMPILe command in BTEQ to create stored procedures in the database.
Line Numbers in Files

The editor that you use to create BTEQ scripts or run files must not add line numbers to the file that you create. The editor can provide line numbers on your display screen, but if they are included in the file that you create, BTEQ interprets them as part of your input.

Command Execution Order

When creating BTEQ scripts, run files, or macros, use the BTEQ commands and Teradata SQL statements in the same order that you want them executed, as you would in an interactive session.

Creating and Using Scripts

A script is a special input file that contains BTEQ commands and Teradata SQL, and is defined as the SYSIN file for automatic execution when you invoke BTEQ.

Creating a Script

To create a BTEQ script to display the contents of table Department in your default database, create a file called SAMPFILE, and enter the following BTEQ script:

```
.LOGON tdpid/userid,password
.SET separator '|
SELECT * FROM department;
.LOGOFF
.EXIT
```

Identifying the Script as the Standard Input File

After creating and saving a script file you must identify it as the standard input or SYSIN file before you execute BTEQ.

<table>
<thead>
<tr>
<th>To identify the script...</th>
<th>Use...</th>
</tr>
</thead>
<tbody>
<tr>
<td>on channel-attached systems</td>
<td>the BTEQ TDSBTEQ JCL procedure or a DD statement in a standard JCL file.</td>
</tr>
<tr>
<td>on network-attached systems*a</td>
<td>the following command at your system prompt: BTEQ &lt; SAMPFILE where &lt; identifies SAMPFILE as the standard input file.</td>
</tr>
</tbody>
</table>

a. The use of the -c and -e command line options or the "charset_id" value in the clispb.dat file is highly recommended when a Unicode input file is used. Otherwise, BTEQ will try to interpret the input file as Unicode and set the session character set accordingly, as follows. BTEQ will check the input file for a UTF16 or UTF8 BOM and will automatically change the session character set to the appropriate Unicode encoding. The endianness of a UTF16 BOM must match that of the machine BTEQ is running on. If a BOM-less UTF16 input file is used, BTEQ will validate the first character and will automatically change the session character set to UTF16. Note that BTEQ will not automatically change the session character set for a BOM-less UTF8 input file. If none of the above apply, BTEQ will assume the input file is not Unicode and will default the session character set to ASCII.
Displaying Script Results

Depending on the configuration of your system SYSPRINT specification or system output settings, BTEQ either displays the results on your terminal or writes them to the specified file:

```
.Logon e/fml, password, acctid
*** Logon successfully completed.
*** Total elapsed time was 3 seconds.

.SET SEPARATOR ' | '
SELECT * FROM department;

*** Query completed. 5 rows found. 4 columns returned.
*** Total elapsed time was 3 seconds.

DeptNo | DeptName  | Loc |  MgrNo
------  --------------  ---  -----  
500 | Engineering  | ATL |  10012
700 | Marketing  | NYC |  10021
300 | Exec Office  | NYC |  10018
600 | Manufacturing | CHI |  10007
100 | Administration| NYC |  10011

.LOGOFF
*** You are now logged off from the DBC.
.EXIT;
```

Omitting Your Userid and Password from the Script File

If, for security reasons, you do not want to include the logon string with your userid and password in the script file, you can enter them on the command line when you invoke BTEQ. For example, if you deleted the logon line from your SAMPFILE script, to invoke BTEQ and run the script you would enter:

```
BTEQ .LOGON tdpid/userid,password
```

Creating and Using Run Files

BTEQ run files are the same as scripts or input stream files except that they are not defined as the SYSIN file for automatic execution when you invoke BTEQ. Instead, you must define the file with a z/OS DD statement. Then, invoke BTEQ and use the RUN command to execute the file.

Creating a Run File

To create a BTEQ run file to display the contents of a table `Department` in your default database, create a file called SAMPFILE, and enter the following BTEQ script:

```
.LOGON tdpid/userid,password
SELECT * FROM department;
.LOGOFF
```
Chapter 3: Using BTEQ
Using Scripts, Run Files, Macros, and Stored Procedures

Identifying a Run File as a Valid ddname
After creating and saving your run file, identify it as a valid ddname:

<table>
<thead>
<tr>
<th>If Using</th>
<th>Enter</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSO</td>
<td>ALLOC FI(SAMPFILE) DA(sampfile.script) SHR</td>
</tr>
</tbody>
</table>

Executing a Run File
To execute the run file, enter either form of the BTEQ RUN command:

```
.RUN FILE=sampfile
```
or

```
.RUN DDNAME=sampfile
```

Before running your file, BTEQ echoes the RUN command, followed by the BTEQ command prompt. BTEQ then executes each statement of your run file, just as though they were entered interactively.

Terminating a Run File
If your run file does not end with either an EXIT or a QUIT command, BTEQ returns the following warning message, followed by the BTEQ command prompt, after executing the last statement:

```
*** Warning:  EOF on INPUT stream.
BTEQ -- Enter your Teradata SQL request or BTEQ command:
```

Note: For information on I/O errors and abends, refer to “I/O Errors and Abends” on page 92.

Displaying Run File Results
Depending on the configuration of your SYSPRINT specification, BTEQ either displays the results on your terminal or writes them to the specified file:

```
.LOGON fml,
*** Logon successfully completed.
*** Total elapsed time was 3 seconds.

SELECT * FROM department;
*** Query completed. 5 rows found. 4 columns returned.
*** Total elapsed time was 3 seconds.

DeptNo  DeptName  Loc  MgrNo
-------  ------------  ---  -----
500  Engineering  ATL  10012
700  Marketing  NYC  10021
300  Exec Office  NYC  10018
600  Manufacturing CHI  10007
100  Admin  NYC  10011

.LOGOFF
*** You are now logged off from the DBC.
*** Warning: EOF on INPUT stream.

Note: Although you may want to execute a run file interactively and display the results on your terminal screen for short jobs, for longer jobs, you may want BTEQ to write the results to a SYSPRINT file. In those cases, you must also define your SYSPRINT file as a valid ddname using a z/OS DD statement before you execute the RUN command.

Revising Scripts and Run Files

There are two ways to edit BTEQ scripts and run files:

<table>
<thead>
<tr>
<th>If you are...</th>
<th>Then use...</th>
</tr>
</thead>
<tbody>
<tr>
<td>logged on to BTEQ</td>
<td>the TSO command to access your channel-attached system editor, or the OS command to access your network-attached system editor.</td>
</tr>
<tr>
<td>logged off BTEQ</td>
<td>any available system editor.</td>
</tr>
</tbody>
</table>

Note: You cannot use an editor that introduces line numbers into the file. When editing a script or run file from within BTEQ, you must have sufficient memory allocated to your BTEQ session.

Accessing the System Editor

To access your system editor from within BTEQ:

<table>
<thead>
<tr>
<th>To access the system editor on...</th>
<th>Enter...</th>
</tr>
</thead>
<tbody>
<tr>
<td>z/OS</td>
<td>the TSO command: .TSO ISPF</td>
</tr>
<tr>
<td>UNIX</td>
<td>the OS command as either: .OS VI filename or .OS ED filename</td>
</tr>
<tr>
<td>Windows</td>
<td>the OS command as either: .OS NOTEPAD filename or .OS EDLIN filename</td>
</tr>
</tbody>
</table>

Testing Changes to Files

After revising your file, exit the system editor to return to BTEQ and use the RUN command to verify your changes.
Creating and Using Macros

Additional Database Privileges
You must have the following additional database privileges, which are granted by the owner of the database, to create or use a macro:

- The CREATE MACRO privilege.
- Access to the appropriate database elements. (You cannot create a macro to access a database table that you do not have permission to access.)

Also, after you have created a macro, you must grant EXECUTE privileges to any other users before they can execute your macro.

Executing Commands
BTEQ executes the commands and SQL statements in the same order that you enter them. You must enclose each BTEQ command in a Teradata SQL ECHO statement. Then, when you execute the macro, Teradata Database executes the Teradata SQL statements and returns the contents of the ECHO statements (your BTEQ commands) to BTEQ.

Warning: It is suggested that you echo BTEQ commands only in Field Mode. You will not get the behavior you might expect when you attempt to echo commands to BTEQ using other response modes. You will not receive error messages, plus the echoed text may be dumped rather than executed.

Creating a Macro
To create a macro, enter the Teradata SQL CREATE MACRO and ECHO commands in an interactive BTEQ session as follows:

```
CREATE MACRO macroname AS
(ECHO '.BTEQcommand'
 ;Teradata SQLrequest;
);
```

The distribution of Teradata SQL requests and BTEQ commands across the command line is arbitrary. The following syntax, for example, performs the same function, but requires additional punctuation:

```
CREATE MACRO macroname AS
(ECHO '.BTEQ command'
 ;Teradata SQLrequest';);
```

Available Commands for Teradata SQL Macros

Table 1: BTEQ Commands for Teradata SQL Macros

<table>
<thead>
<tr>
<th>BTEQ Command</th>
<th>Available for Teradata SQL Macro</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABORT</td>
<td>No</td>
</tr>
<tr>
<td>AUTOKEYRETRIEVE</td>
<td>Yes</td>
</tr>
<tr>
<td>COMPILE</td>
<td>Yes</td>
</tr>
<tr>
<td>DECIMALDIGITS</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Table 1: BTEQ Commands for Teradata SQL Macros (continued)

<table>
<thead>
<tr>
<th>BTEQ Command</th>
<th>Available for Teradata SQL Macro</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULTS</td>
<td>Yes</td>
</tr>
<tr>
<td>ECHOREQ</td>
<td>No</td>
</tr>
<tr>
<td>ENCRYPTION</td>
<td>Yes</td>
</tr>
<tr>
<td>ERRORLEVEL</td>
<td>Yes</td>
</tr>
<tr>
<td>ERROROUT</td>
<td>Yes</td>
</tr>
<tr>
<td>EXIT</td>
<td>No</td>
</tr>
<tr>
<td>EXPORT</td>
<td>No</td>
</tr>
<tr>
<td>EXPORETEJECT</td>
<td>Yes</td>
</tr>
<tr>
<td>FOLDLINE</td>
<td>Yes</td>
</tr>
<tr>
<td>FOOTING</td>
<td>Yes</td>
</tr>
<tr>
<td>FORMAT</td>
<td>Yes</td>
</tr>
<tr>
<td>FORMCHAR</td>
<td>Yes</td>
</tr>
<tr>
<td>FULLYEAR</td>
<td>Yes</td>
</tr>
<tr>
<td>GOTO</td>
<td>No</td>
</tr>
<tr>
<td>HALT EXECUTION</td>
<td>No</td>
</tr>
<tr>
<td>HANG</td>
<td>Yes</td>
</tr>
<tr>
<td>HEADING</td>
<td>Yes</td>
</tr>
<tr>
<td>HELP BTEQ</td>
<td>Yes</td>
</tr>
<tr>
<td>IF...THEN...</td>
<td>Yes</td>
</tr>
<tr>
<td>IMPORT</td>
<td>No</td>
</tr>
<tr>
<td>INDICDATA</td>
<td>Yes</td>
</tr>
<tr>
<td>LABEL</td>
<td>No*</td>
</tr>
<tr>
<td>LARGEDATAMODE</td>
<td>Yes</td>
</tr>
<tr>
<td>LOGDATA</td>
<td>No</td>
</tr>
<tr>
<td>LOGMECH</td>
<td>No</td>
</tr>
<tr>
<td>LOGOFF</td>
<td>No</td>
</tr>
<tr>
<td>LOGON</td>
<td>No</td>
</tr>
<tr>
<td>LOGONPROMPT</td>
<td>Yes</td>
</tr>
<tr>
<td>MAXERROR</td>
<td>Yes</td>
</tr>
<tr>
<td>NOTIFY</td>
<td>No</td>
</tr>
<tr>
<td>BTEQ Command</td>
<td>Available for Teradata SQL Macro</td>
</tr>
<tr>
<td>--------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>NULL</td>
<td>Yes</td>
</tr>
<tr>
<td>OMIT</td>
<td>Yes</td>
</tr>
<tr>
<td>OS</td>
<td>Yes</td>
</tr>
<tr>
<td>PACK</td>
<td>Yes</td>
</tr>
<tr>
<td>PAGEBREAK</td>
<td>Yes</td>
</tr>
<tr>
<td>PAGELENGTH</td>
<td>Yes</td>
</tr>
<tr>
<td>QUIET</td>
<td>Yes</td>
</tr>
<tr>
<td>QUIT</td>
<td>No</td>
</tr>
<tr>
<td>RECORDMODE</td>
<td>Yes</td>
</tr>
<tr>
<td>REMARK</td>
<td>Yes</td>
</tr>
<tr>
<td>REPEAT</td>
<td>No</td>
</tr>
<tr>
<td>REPEATSTOP</td>
<td>Yes</td>
</tr>
<tr>
<td>REPORTALIGN</td>
<td>Yes</td>
</tr>
<tr>
<td>RETCANCEL</td>
<td>Yes</td>
</tr>
<tr>
<td>RETLIMIT</td>
<td>Yes</td>
</tr>
<tr>
<td>RETRY</td>
<td>Yes</td>
</tr>
<tr>
<td>RTITLE</td>
<td>Yes</td>
</tr>
<tr>
<td>RUN</td>
<td>No</td>
</tr>
<tr>
<td>SEPARATOR</td>
<td>Yes</td>
</tr>
<tr>
<td>SESSION CHARSET</td>
<td>Yes</td>
</tr>
<tr>
<td>SESSION RESPBUFLEN</td>
<td>Yes</td>
</tr>
<tr>
<td>SESSION SQLFLAG</td>
<td>No*</td>
</tr>
<tr>
<td>SESSION TRANSACTION</td>
<td>No*</td>
</tr>
<tr>
<td>SESSION TWORESPBUFS</td>
<td>Yes</td>
</tr>
<tr>
<td>SESSIONS</td>
<td>No*</td>
</tr>
<tr>
<td>SHOW CONTROLS</td>
<td>Yes</td>
</tr>
<tr>
<td>SHOW ERRORMAP</td>
<td>Yes</td>
</tr>
<tr>
<td>SHOW VERSIONS</td>
<td>Yes</td>
</tr>
<tr>
<td>SIDETITLES</td>
<td>Yes</td>
</tr>
<tr>
<td>SKIPDOUBLE</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Although it is possible to ECHO this command, the command will fail as a result of other constraints. For example, all sessions must be logged off for you to use the SESSION command.

**Macro Example**

To create a macro named `deptdisplay`, for example, that includes the following BTEQ SEPARATOR command and Teradata SQL SELECT statement:

```
.SET SEPARATOR ' | '
SELECT * FROM department;
```

enter:

```
CREATE MACRO deptdisplay AS
(ECHO '.SET SEPARATOR '' | ''
;SELECT * FROM department;
);
```

Note that when you enclose single quotes in quotes, you double the inner quotes, and each inner single quote becomes two single quotes.

After creating your macro, BTEQ displays:

```
*** Macro has been created.
*** Total elapsed time was 3 seconds.
```

**Note:** Develop more complex macros by placing them in a file and running the file in a BTEQ batch job.

**Granting the Macro EXECUTE Privilege**

As the creator of a macro, you automatically receive the execute privilege on the macro. To successfully execute the macro, however, you must have been granted access privileges to all of the objects named in the macro. For example, to execute the `deptdisplay` macro that you created in the prior subsection, you must have the select privilege on the Department table.

### Table 1: BTEQ Commands for Teradata SQL Macros (continued)

<table>
<thead>
<tr>
<th>BTEQ Command</th>
<th>Available for Teradata SQL Macro</th>
</tr>
</thead>
<tbody>
<tr>
<td>SKIPLINE</td>
<td>Yes</td>
</tr>
<tr>
<td>SUPPRESS</td>
<td>Yes</td>
</tr>
<tr>
<td>TDP</td>
<td>Yes</td>
</tr>
<tr>
<td>TIMEMSG</td>
<td>Yes</td>
</tr>
<tr>
<td>TITLEDASHES</td>
<td>Yes</td>
</tr>
<tr>
<td>TSO</td>
<td>Yes</td>
</tr>
<tr>
<td>UNDERLINE</td>
<td>Yes</td>
</tr>
<tr>
<td>WIDTH</td>
<td>Yes</td>
</tr>
<tr>
<td>=</td>
<td>No</td>
</tr>
</tbody>
</table>

* Although it is possible to ECHO this command, the command will fail as a result of other constraints. For example, all sessions must be logged off for you to use the SESSION command.
Similarly, anyone that you grant permission to execute the deptdisplay macro must also have the same privileges.

Use the Teradata SQL GRANT statement to allow other users to execute your macro:

```
GRANT EXECUTE ON macroname TO userid;
```

To grant the execute privilege to user jnw on the macro deptdisplay, enter:

```
GRANT EXECUTE ON deptdisplay TO jnw;
```

**Executing Macros**

You can execute macros either interactively, or include them in BTEQ scripts or input stream files.

To execute a macro interactively, use the Teradata SQL EXECUTE statement:

```
EXECUTE macroname;
```

To execute the macro deptdisplay, enter:

```
EXECUTE deptdisplay;
```

**Including a Macro in a Script**

Use the editor on your system to include a BTEQ macro in script or input stream file. To execute the deptdisplay macro from a script, for example, enter the Teradata SQL EXECUTE statement after the LOGON statement in the script:

```
.LOGON tdpid/userid, password
EXECUTE deptdisplay;
.LOGOFF
```

**Creating Reports with Macros**

You can create a report from a macro by instructing BTEQ in your prepared script or input stream file to export the data returned as a report. To do this, you can either:

- Create your macro on Teradata Database and execute it from the script file. In this case, the macro would contain the appropriate BTEQ commands to generate a report, and your script file would contain the BTEQ EXPORT command, the Teradata EXECUTE macro statement, and other BTEQ commands. This is the preferred method.

- Place your macro in the script file and create and execute it from there. This method creates the macro every time it executes the request, a major disadvantage.

**Macro Example**

The following is an example of a macro to produce a report called Department Information:

```sql
CREATE MACRO deptdisplay AS (
  ECHO '.SET SEPARATOR " | " ;
  ECHO '.SET RTITLE "Department Information" ;
  ECHO '.SET FORMAT ON;
  SELECT * FROM department;)
```
Defining the ddname of the Export File

After creating the macro, use the appropriate syntax to define the ddname of the export file as follows:

<table>
<thead>
<tr>
<th>If Using</th>
<th>Enter</th>
</tr>
</thead>
<tbody>
<tr>
<td>z/OS</td>
<td>the following JCL:</td>
</tr>
<tr>
<td></td>
<td>//EXPORT EXEC TDSBTEQ,</td>
</tr>
<tr>
<td></td>
<td>// L='logon string'</td>
</tr>
<tr>
<td></td>
<td>//REPORT DD DSN=your.report.dataname,DISP=(,KEEP),</td>
</tr>
<tr>
<td></td>
<td>// DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160),</td>
</tr>
<tr>
<td></td>
<td>// SPACE=(TRK,(5,5))</td>
</tr>
<tr>
<td></td>
<td>//SYSIN DD DISP=SHR,DSN=your.BTEQ.input</td>
</tr>
</tbody>
</table>

Note: On z/OS, if you use the TSO command to define the ddname of the export file in the input stream file, you do not need to define the ddname in the CLIST. Your input stream, for example, might include a command similar to the following:

```ts
.TSO ALLOCATE DDNAME(repfile) DSNAME(your.report.dataname) SHR
```

An ALLOCATE statement must be used for TSO.

Running the Macro

With the macro created on Teradata Database and the export file defined, the following script exports the data returned as a report and executes the macro:

```ts
.LOGON tdpid/userid, password
.EXPORT REPORT DDNAME=report
EXECUTE deptdisplay;
.SET FORMAT OFF
.EXPORT RESET
.LOGOFF
```

Execute the z/OS JCL (from outside of BTEQ) to submit the script and execute the macro.

Report Example

88/03/07 Department Information Page 1

<table>
<thead>
<tr>
<th>DeptNo</th>
<th>DeptName</th>
<th>Loc</th>
<th>MgrNo</th>
</tr>
</thead>
<tbody>
<tr>
<td>------</td>
<td>--------------</td>
<td>-----</td>
<td>-------</td>
</tr>
<tr>
<td>500</td>
<td>Engineering</td>
<td>ATL</td>
<td>10012</td>
</tr>
<tr>
<td>700</td>
<td>Marketing</td>
<td>NYC</td>
<td>10021</td>
</tr>
<tr>
<td>300</td>
<td>Exec Office</td>
<td>NYC</td>
<td>10018</td>
</tr>
<tr>
<td>600</td>
<td>Manufacturing</td>
<td>CHI</td>
<td>10007</td>
</tr>
<tr>
<td>100</td>
<td>Administration</td>
<td>NYC</td>
<td>10011</td>
</tr>
</tbody>
</table>

Command Execution Messages

The sample macro would produce the following command execution messages in the SYSPRINT file:

```ts
.LOGON e/fml,
```
*** Logon successfully completed.
*** Total elapsed time was 3 seconds.

.EXPORT REPORT FILE=report.out
*** To reset export, type .EXPORT RESET

.SET RTITLE 'Department Information'
.SET FORMAT ON
EXECUTE deptdisplay;
*** Echo accepted.
*** Total elapsed time was 3 seconds.
*** Query completed. 5 rows found. 4 columns returned.

.SET FORMAT OFF
.EXPORT RESET
*** Output returned to console.

.LOGOFF
*** You are now logged off from the DBC.
.EXIT

You can discard the SYSPRINT file after examining it.

Creating and Executing a Macro from a Script

The following example shows how to produce the same report by creating and executing the
macro from the script file:

.LOGON e/fml,notebook
.EXPORT REPORT DDNAME=report
CREATE MACRO deptdisplay AS ( ECHO '.SET SEPARATOR'; ECHO '.SET RTITLE "Department Information" '; ECHO '.SET FORMAT ON'; SELECT * FROM department;);
EXECUTE deptdisplay;
.SET FORMAT OFF
.EXPORT RESET
/logout

Creating and Using Stored Procedures

Database Privileges Required

You must have the following additional database privileges to create or use a stored procedure:

- CREATE PROCEDURE privilege
- Access to the appropriate database elements. (You cannot create a procedure that
  references a database table for which you do not have access permission.)

To replace or modify an existing procedure, you must have the DROP privilege on the
procedure or DROP PROCEDURE privilege on the database containing it.

Also, after you have created a procedure, you must grant EXECUTE privilege on the
procedure or EXECUTE PROCEDURE privilege on that database that contains it, to any
other users before they can execute the procedure.
Note: The EXECUTE privilege allows users to execute macros, while EXECUTE
PROCEDURE is specific to stored procedures.

Types of Stored Procedures
BTEQ supports external and non-external stored procedures.

An external stored procedure is defined with a CREATE PROCEDURE or REPLACE
PROCEDURE statement, along with EXTERNAL references for source files.

For example:

```
CREATE PROCEDURE xyz
  (IN data1 INTEGER, OUT data2 INTEGER)
  LANGUAGE CPP NO SQL PARAMETER STYLE TD_GENERAL
  EXTERNAL NAME 'CS!xyz!source.cpp';
```

The C or C++ files can reside on the server or the client. The server will call for the transfer of
client-resident files as needed to satisfy input requirements for the CREATE PROCEDURE or
REPLACE PROCEDURE request. Source files must be encoded as ASCII (workstation) or
EBCDIC (mainframe), no matter what the current session character set is. See SQL External
Routine Programming for more information.

The retrieval of client-resident external stored procedure files is handled the same way that
User-Defined Function files are handled. See “Specifying Files for User-Defined Functions”
for more details.

A non-external stored procedure is created or replaced using the COMPILE command, which
takes a single Stored Procedure Language (SPL) file as input. The encoding of this file must
match that of the current session character set.

The remainder of this section provides details for using regular non-external stored
procedures.

Command Execution
Specify the following command to create stored procedures on the Teradata server:

```
.COMPILE FILE = example.spl;
```

As BTEQ processes the COMPILE command, it uses the specified file's content as the stored
procedure source text input. It sends the stored procedure source text to the Teradata server
for processing the stored procedure definition.

Creating or Replacing a Stored Procedure
To create a stored procedure, invoke the text editor on your system, create a file named
example.spl, and enter the stored procedure source text:

```
CREATE PROCEDURE samplespl ()
BEGIN
  /* SPL Statements*/
  DECLARE V1 INTEGER;
  SET V1 = 100;
END;
```

The preceding example creates a procedure without any parameters.
To replace this with another stored procedure with parameters, create a file named example1.spl, and enter the stored procedure source text:

```
REPLACE PROCEDURE samplesp1 (IN pAccountNo INTEGER, OUT pAmount DECIMAL (10,2))
BEGIN
    /* SPL Statements*/
    SELECT : Amount INTO : pAmount FROM : Account1 WHERE aCCTnO = :pAccountNo;
END;
```

Submit the COMPILE command with this file name.

**Executing Stored Procedures**

You can execute stored procedures either interactively, or include them in BTEQ scripts or input stream files.

Use the Teradata SQL CALL statement to execute a stored procedure interactively:

```
call samplesp1 (8888, pAmount);
```

If the procedure is created with parameters, you must specify the parameter arguments within the parenthesis.

**Including a Stored Procedure in a Script**

Use the text editor on your system to include a stored procedure in a BTEQ script or input stream file. To execute the procedure samplesp1 from a script, for example, enter the Teradata SQL CALL statement after the LOGON statement in the script:

```
.logon tdpid/userid, password
CALL samplesp1 (8888, pAmount);
.logoff
```

**Command Execution Messages**

If you execute the sample stored procedure `act`, specified in the file `example.spl`, and containing one parameter, BTEQ produces the following command execution messages in the SYSPRINT file:

```
.logon mydbs/tdsp01
Password:
*** Logon successfully completed.
*** Transaction Semantics are BTET.
*** Character Set Name is 'ASCII'.
*** Total elapsed time was 1 second.

BTEQ -- Enter your SQL request logon or BTEQ command:
.compile file example.spl

*** Procedure has been created. .
*** Total elapsed time was 5 seconds.

BTEQ -- Enter your SQL request logon or BTEQ command:
call samplesp1 (8888, pAmount);
```
*** Procedure has been executed.  
*** Total elapsed time was 1 second.

pAmount
--------
10000.00

BTEQ -- Enter your SQL request logon or BTEQ command:
  sel * from account1;

*** Query completed. One row found. 2 columns returned.  
*** Total elapsed time was 1 second.

accountno amount
-------- --------
8888      10000.00

.LOGOFF
*** You are now logged off from the DBC.

Creating and Executing a Stored Procedure from a Script
The following example shows how to create and execute a stored procedure from the script file, assuming that the script is running in ANSI mode. The example.spl file contains the stored procedure source text.

.LOGON e/fml,notebook
.COMPILE FILE = example.spl;
COMMIT;
CALL samplesp1 (8888, pAmount);
.LOGOFF

The next example is a script for a session running in Teradata mode:

.LOGON e/fml,notebook
BT;
.COMPILE FILE = example.spl;
ET;
CALL samplesp1 (pAccountNo, 10000.00);
.LOGOFF

Testing and Branching

When writing BTEQ scripts to run in batch mode, you can use commands to test conditions and respond to a condition by skipping ahead to another portion of the script.

Testing and Branching Commands

There are two commands that perform these functions:
IF... THEN... Command Process

The BTEQ IF...THEN... command is a conventional condition-testing command. After testing the condition stated in the IF clause, BTEQ resumes command execution as follows:

<table>
<thead>
<tr>
<th>If</th>
<th>Then</th>
</tr>
</thead>
<tbody>
<tr>
<td>the condition stated in the IF clause is TRUE</td>
<td>BTEQ executes the command stated in the THEN clause.</td>
</tr>
<tr>
<td>the condition stated in the IF clause is FALSE</td>
<td>BTEQ disregards the command stated in the THEN clause and executes the next command.</td>
</tr>
</tbody>
</table>

Note: The testing always refers to the outcome of the most recent Teradata SQL request, the one that immediately precedes the IF...THEN... command. You cannot branch based on the outcome of an earlier request.

GOTO Command Process

The BTEQ GOTO command is a straightforward branching command that you can use with the THEN clause of the IF...THEN... command to provide the destination of the branch operation.

Enter the GOTO command with a `labelname` parameter, which designates skip destination. Identify the skip destination with a LABEL command.

Note: The GOTO command only branches forward; you cannot branch back to a statement listed before the GOTO command. When you use the GOTO command, the LABEL command that identifies the `labelname` must appear later in your script.

GOTO Labelname and LABEL Command

The GOTO command provides the `labelname` attribute that identifies the destination of the branch.

Only the first ten characters of the attribute are significant. Though you can use longer names, always avoid specifications such as:

```plaintext
.LABEL experiment1
.LABEL experiment2
```

Because only the first ten characters are valid, BTEQ would branch to the first occurrence of .LABEL experiment, in response to:

```plaintext
.GOTO experiment2
```
Testing Status Values

BTEQ retains three status values reflecting the results of the most recent Teradata SQL request. These are the values that you can test with the IF clause of an IF...THEN... command:

<table>
<thead>
<tr>
<th>Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERRORCODE</td>
<td>Indicates the actual completion code associated with the request.</td>
</tr>
<tr>
<td>ERRORLEVEL</td>
<td>Indicates the severity level associated with an error code.</td>
</tr>
<tr>
<td>ACTIVITYCOUNT</td>
<td>Indicates the actual number of rows affected by the request. For a request resulting in a selection of rows, this will be the number of rows actually returned to BTEQ from the database.</td>
</tr>
</tbody>
</table>

Note: An ERRORCODE value of zero signifies a request that completed without error.

Testing Example

For example, if a Teradata SQL SELECT request succeeded and returned the contents of 23 rows, the ERRORCODE value associated with the request would be zero, and the ACTIVITYCOUNT would be 23. The value of ERRORLEVEL depends upon whether the submitted request results in an error or not. Or, if a Teradata SQL DELETE request encountered no rows that met the specified delete criteria, but encountered no error conditions, all three values would be zero—no error condition, consequently no severity level, and no rows affected.

Note: The value of ERRORLEVEL does not reset to zero as is done for ERRORCODE when a submitted request does not result in an error. If the value of ERRORLEVEL needs to be tested against the last request submitted, ERRORCODE should be tested first, to see whether the request was successful or not.

For more information about error codes and severity levels, refer to “Handling Errors” on page 85. Table 2, Table 3, and Table 4 list the Teradata Database error messages that are assigned default severity levels (return codes) of 04, 08, and 012, respectively. For a complete list of all Teradata Database error codes, refer to Messages.

Testing Conditions

The six conditions that you can test for are:

<table>
<thead>
<tr>
<th>Condition</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>equal to</td>
</tr>
<tr>
<td>&lt;&gt; , !=, ~=, or ^=</td>
<td>not equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>greater than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>greater than or equal to</td>
</tr>
<tr>
<td>&lt;</td>
<td>less than</td>
</tr>
</tbody>
</table>
Producing Conditional Messages

One common use of BTEQ testing and branching is to produce a message if certain conditions are met. For example, your script may include a request that you expect to actually change the database. In this case, an activity count of zero would indicate a problem, even if there was no error code returned. If the request was critical to your operations, you might include the following command in your script, right after the request:

```
.IF ACTIVITYCOUNT = 0 THEN .QUIT 65
```

In this case, you would have predefined the return code of 65 to indicate that the request did not change the database.

Or, before quitting BTEQ, you could branch to a different part of the script and use the REMARK command to specify a descriptive message to elaborate on the return code, as follows:

```
.IF ACTIVITYCOUNT = 0 THEN .GOTO PROB65
.
.
.LOGOFF
.EXIT 0
.LABEL PROB65
.REMARK 'Prob 65: The . . . request did not affect any rows'
.QUIT 65
```

Using the REPEAT Command

Frequently, when updating Teradata Database, you will need to make the same change to a large number of rows. When updating multiple rows, it is helpful to use some means of eliminating the repetitive work required to update them individually. The REPEAT command serves this purpose.

Updating Rows Using a Single Request

If you can describe all of the rows by some formula, then you can use a single request to update them all. The following request, for example, updates the salaries of all employees in department 600 of the database Personnel:

```
update personnel.employee set salary=salary*1.07
where deptno=600;
```

Updating Rows Using Multiple Requests

If you cannot describe all of the rows by a formula, you must describe them individually. The following requests, for example, update the salaries of two employees:
update personnel.employee set salary=salary*1.07
where empno = 10006;

update personnel.employee set salary=salary*1.07
where empno = 10013;

This approach is convenient for a few updates, but if there were hundreds, the script would become very long. And, it would be very inflexible. You would have to edit each entry to use the script again for another list of employees.

**Updating Rows by Importing a File**

A better approach would be to use your system editor to create a separate file (for example, RAISEEMP) containing only the employee numbers 10006 and 10013. Then use the BTEQ IMPORT command with a REPEAT or = command to update the rows in the database.

**Using the REPEAT Command when Importing a File**

The REPEAT command appears before the request and specifies the total number of requests to be submitted.

The value for the REPEAT command is * by default, which repeats the next request as many times as it takes to exhaust the data in the import file.

The following BTEQ script, using the BTEQ REPEAT command, would open the character-format file RAISEEMP, and repeat the update sequence twice:

```
.import data file=raiseemp
.repeat 2
using enumb (char(5))
.update personnel.employee set salary=salary*1.07
where empno = :enumb;
```

For each employee, BTEQ would read a value from the RAISEEMP file for the variable called enumb and carry out the UPDATE for the row whose EmpNo equals the value of enumb.

**Caution:** When using a file, originally exported using EXPORT command of BTEQ, as the source for IMPORT command across a different platform type, ensure that the endianness type of both platforms is the same. This can be verified from the “Client Platform Byte Order” tab in the output of SHOW CONTROLS command.

**Using the = Command When Importing a File**

The = command appears after the request and specifies one less than the total number of requests to be submitted, because the first request has already been submitted when the = command is stated.

The = command is used most often interactively to resubmit a SELECT statement that had run under the wrong formatting option. After obtaining output in an unexpected format, you can submit the BTEQ commands to correct the formatting options and then use the = command to get the reformatted output.

The value for the = command is 1 by default, which repeats the prior command one time only. For example:
Chapter 3: Using BTEQ
Using the REPEAT Command

80 Basic Teradata Query Reference

.REPEAT 7
=6

The following script, using the BTEQ = command would perform the same update:

.IMPORT data file=raiseemp
using enumb (char(5))
update personnel.employee set salary=salary*1.07
where empno = :enumb;
=1

The choice between using the REPEAT command and the = command is simply one of personal preference. The only substantial differences between the two commands are in the value for the count field, and the default values.

Caution: Always be very careful when using the = command with a request containing other than a SELECT statement, which can alter the database more than once. (SELECT statements do not actually alter the database.)

Using REPEAT with Multiple Sessions

The REPEAT command is beneficial when working with multiple rows and/or multiple sessions. The default database can be specified within the Teradata SQL statement, as with the Teradata SQL UPDATE statement in prior examples. Or, enter the REPEAT command with the number of sessions before the Teradata SQL DATABASE statement, which is another way to define the default database.

To run two concurrent sessions that update the Employee table in the Personnel database, for example, enter:

.SET sessions 2
.REPEAT 2
database personnel;
.IMPORT data file=raiseemp
.REPEAT 2
using enumb (char(5))
update employee set salary=salary*1.07
where empno = :enumb;

The first REPEAT command specifies the default database for both sessions. The second REPEAT command runs the two sessions until the entire table is updated. Without specifying the first REPEAT command, only the first session would use the Personnel database; the other session would not have a defined database.

The following example reflects the use of the REPEAT command to ensure multiple sessions are established with the same query band. For the following example, propagation of a set query band is not automatic across sessions.

.SET SESSIONS 3
.LOGON MyDB/MyID,MyPW;
.REPEAT 3
SET QUERY_BAND= 'MyNAME=12345;' FOR SESSION;
Interrupting and Aborting Requests

You can interrupt or abort BTEQ processing using either the Break key or the ABORT command.

Using the Break Key on Workstations

Use the Break key to perform the following functions:

<table>
<thead>
<tr>
<th>If you want to...</th>
<th>Then use...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interrupt BTEQ</td>
<td>the single-break function.</td>
</tr>
<tr>
<td>Abort the previous request</td>
<td>the double-break function.</td>
</tr>
<tr>
<td>Terminate BTEQ</td>
<td>the triple-break function.</td>
</tr>
</tbody>
</table>

For more information about aborting BTEQ requests, refer to “ABORT” on page 130.

The break function has the following syntax:

```
<break>
```

where

<break> represents pressing the CTRL+C keys on your workstation keyboard.

Single-Break

The single-break function interrupts BTEQ while it is processing a previous request. BTEQ displays:

BREAK Received. Input command:

Double-Break

At the input command prompt, you can either press the Break key again (the double-break function), or enter CONTINUE or ABORT.

Enter CONTINUE if you want to recover from the interrupt and resume with BTEQ as if the interrupt never occurred.

Either enter ABORT or press the Break key again if you want to abort the previous request.
**Triple Break**

Enter a third break to terminate BTEQ. Alternatively, entering HX after the first break also terminates BTEQ.

**Note:** The above behavior is for console BTEQ only. Interrupt and Abort functions are available in BTEQWIN on the Sessions menu; CTRL+C is undefined in BTEQWIN.

**Using the Break Key on z/OS**

Use the **Break** key to perform the following function:

<table>
<thead>
<tr>
<th>If you want to...</th>
<th>Then use...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interrupt BTEQ</td>
<td>the single-break function.</td>
</tr>
</tbody>
</table>

For more information about aborting BTEQ requests, refer to “**ABORT**” on page 130.

The break key command has the following syntax:

```
<break_key> <Unlocking_terminal> CONTINUE ABORT HX
```

where:

- `<break_key>` represents pressing the **PA1** key on your mainframe keyboard.
- `<Unlocking_terminal>` comprises the following sequence of steps:
  1. Press **RESET**
  2. Press **PA1**
  3. Press **RESET**
  4. Press **PA1**

**Note:** When BTEQ is in the process of displaying multi-page output for a query, the keyboard is not locked and pressing the **PA1** key sends a user interrupt to BTEQ.

In contrast, when the TSO terminal is waiting for an outstanding request, the keyboard is locked to prevent input. To unlock the terminal keyboard and send a user interrupt to BTEQ, follow the sequence of steps above to unlock the terminal and interrupt BTEQ.
**Single Break**

The single break function interrupts BTEQ while it is processing a previous request. It is generated either by pressing the break key or pressing the sequence of keys that unlocks the terminal. On receipt of a single break, BTEQ displays:

*** PA1 key hit! Enter ‘ABORT’ to abort current operation
*** or ‘HX’ to abort transaction and exit BTEQ:

At the he input command prompt, enter CONTINUE, ABORT, or HX.

Enter **CONTINUE** to recover from the interrupt and resume BTEQ as if the interrupt never occurred.

Enter **ABORT** to abort the previous request.

Enter **HX** to abort the transaction and terminate BTEQ.

**Aborting Requests with the ABORT Command**

You can use the ABORT command to stop a request to Teradata Database, and restore any changes that were caused by the request.

When you issue the ABORT command, Teradata Database:

1. Stops processing the request.
2. Discards all of the responses that it has prepared for the request.
3. Backs out all changes that the request made to the database.
4. Aborts the transaction in which the request was embedded.
5. Returns a failure message, indicating that the request was aborted.

**Timing the ABORT Request**

To successfully abort a Teradata SQL request, your abort request must reach Teradata Database before it returns a response to the original request. If your abort request arrives too late, Teradata Database ignores it; neither the original request nor its transaction is affected.

**Restoring Teradata Database After a Failed ABORT Request**

If the request that you tried to abort did not affect the database, or if the affect was not damaging, then the impact of a failed abort request is minimal. If the request did damage the database, then you can correct the damage as follows:

- If the request that you tried to abort was embedded in a transaction, you can still abort the transaction, which includes backing out the effect of the damaging request, by submitting a Teradata SQL ABORT statement as a normal Teradata SQL request.
- If the request that you tried to abort was not embedded in a transaction, you can submit a second request that has the exact opposite effect of the first, thus returning the database to its state before the damaging request.
Chapter 3: Using BTEQ
Entering Operating System Commands from a BTEQ Session

Aborting a File Redirection
When redirecting a standard input stream file, you can abort or interrupt processing on a client system as follows:

- On network-attached systems, you can use the Break key on your workstation keyboard, as described in “Interrupting and Aborting Requests” on page 81.
- For UNIX and Windows, in response to the double-break function, BTEQ aborts the current Teradata SQL statement and continues reading the rest of the file. The triple-break function aborts the current statement and exits BTEQ.

Note: On network-attached systems, do not use the single-break function to abort a file redirection operation. If you press the Break key while a Teradata SQL statement that extends over two lines is executing, BTEQ sends the second line of the statement to Teradata Database and displays an error message. When you use the single-break function, you must follow it with an ABORT command.

Entering Operating System Commands from a BTEQ Session

The ability to enter an operating system command without logging off Teradata Database or ending your BTEQ session is especially helpful when making interactive requests to Teradata Database or developing and debugging scripts and macros. In these situations, for example, you may want to verify the name of a file, or check the contents of a directory on your client system.

You can use one of three BTEQ commands to accomplish this:

<table>
<thead>
<tr>
<th>If Using</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>network-attached systems</td>
<td>the OS command.</td>
</tr>
<tr>
<td>channel-attached systems</td>
<td>the TSO command.</td>
</tr>
</tbody>
</table>

Entering UNIX Commands

For UNIX on network-attached systems, use the UNIX EXECUTE SHELL command with the BTEQ OS command to display your UNIX command prompt. Then, after you have finished entering UNIX commands, press and hold the Control and D keys to return to BTEQ.

UNIX Command Example

```
.OS [exec sh | sh]
$ pg myfile.one
$ cp oldfile.txt newfile.txt
$ cd draft
$ [<cntrl>/<d> or exit]
BTEQ -- Enter your logon or BTEQ command:
```
**Entering Windows Commands**

For Windows, use the DOS command **COMMAND** (secondary command processor) to exit from BTEQ and the **EXIT** command to return to BTEQ.

**Windows Command Example**

```
.OS command
C> type b:myfile.one
C> dir a:
C> exit
```

**Handling Errors**

This section describes the BTEQ error handling process, and the changes that you can make to meet your specific requirements. BTEQ error handling involves these elements:

<table>
<thead>
<tr>
<th>Element</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teradata Database error codes</td>
<td>Produced by Teradata Database in response to the BTEQ commands and Teradata SQL statements that you execute.</td>
</tr>
<tr>
<td>BTEQ return codes</td>
<td>Produced by BTEQ to indicate the completion status of a task or job step.</td>
</tr>
<tr>
<td>Error severity levels</td>
<td>User-assigned values that extend the capability of return codes.</td>
</tr>
<tr>
<td>Maximum error level</td>
<td>MAXERROR is a user-assigned value that specifies the error severity level at which BTEQ terminates job processing.</td>
</tr>
<tr>
<td>Stored procedure compilation errors</td>
<td>Syntax and semantic errors in the stored procedure source text reported by Teradata Database as part of the SUCCESS/OK parcel.</td>
</tr>
</tbody>
</table>

**Teradata Database Error Codes**

The BTEQ error handling capabilities are based on Teradata Database error codes. These are the standard error codes and messages produced in response to your BTEQ commands and Teradata SQL statements. As a BTEQ user, you cannot change, modify or delete these messages.

For a complete description of all Teradata Database error codes, refer to *Messages* for the current release.

**Possible Data-Loss Condition**

In some cases, data might be lost if sessions are aborted by:

- The Gateway Control Utility DISCONNECT SESSION command
- The Performance Monitor ABORT SESSION command
- BTEQ not attempting to reconnect sessions that have been idle for more than 20 minutes after a Teradata Database restart.
After a Teradata Database restart, BTEQ sends a request or fetch on the active session, then CLI starts the reconnect procedure. If a session is idle, it does not send a request or fetch, and CLI cannot perform the reconnect.

**Note:** The 20-minute period is the default value of a gateway configuration specification. The period might be different for your system.

In these cases, Teradata Database returns error code 8018 (Session id is illegal) or 8055 (The session has been forced off) to BTEQ. This means that the Teradata SQL response has been discarded, and the request and the transaction in which it was embedded have been aborted and backed out. If you are using BTEQ:

- for channel-attached systems, BTEQ does not resubmit the request after a session has been aborted, and associated data may be lost.
- for network-attached systems, if multiple sessions are running, BTEQ resubmits the request through another available session. However, if only one session is logged on, and that session is forced off, the transaction is aborted and its associated data is lost. BTEQ is terminated at this point.

The severity of this condition depends on the nature of the transactions that were running when the session was aborted:

- If an insert request was running, the row data may be lost. In this case, the maximum number of rows lost is equivalent to the number of sessions that received the 8018 error.
- If an update or delete request was running, the integrity of the table may be compromised.

In either case, you have to manually resubmit the aborted request.

### BTEQ Return Codes

Return codes are two-digit values that BTEQ returns to your client operating system after completing each job or task. The value of the return code indicates the completion status of the job or task as follows:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Job completed with no errors.</td>
</tr>
<tr>
<td>02</td>
<td>User alert to log on to Teradata Database.</td>
</tr>
<tr>
<td>04</td>
<td>Warning error.</td>
</tr>
<tr>
<td>08</td>
<td>User error.</td>
</tr>
<tr>
<td>12</td>
<td>Severe internal error.</td>
</tr>
</tbody>
</table>

The value of the return code is determined by the error code that BTEQ receives from Teradata Database. Initially, each Teradata Database error code is assigned a default return code value (as shown in the following tables) for return codes of 04, 08, and 12. If, for example, your BTEQ request produces a Teradata Database error 3737–Name is longer than 30 characters–BTEQ sends an 04 return code to your client operating system.
The value of the return code always reflects the most severe error that was encountered.

BTEQ also maintains the return code value as an internal ERRORLEVEL attribute that you can use in subsequent conditional statements. Using return codes in this manner makes it easier for you to determine an appropriate course of action based on a small number of return codes instead of a large number of individual error codes.

**Note:** You can use the BTEQ ERRORLEVEL command to change the return code assignment of any Teradata Database error code, or add additional levels of error severity, as described in the following subsection.

Any Teradata Database message not contained in the following tables returns a condition code (return code) of 8, unless you use the ERRORLEVEL command to override the default, as follows:

```
.SET ERRORLEVEL UNKNOWN SEVERITY n
```

<table>
<thead>
<tr>
<th>Table 2: Return Code 04 Default Error Message Assignments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Error Code</strong></td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>2580</td>
</tr>
<tr>
<td>2667</td>
</tr>
<tr>
<td>3534</td>
</tr>
<tr>
<td>3666</td>
</tr>
<tr>
<td>3737</td>
</tr>
<tr>
<td>3747</td>
</tr>
<tr>
<td>3803</td>
</tr>
<tr>
<td>3804</td>
</tr>
<tr>
<td>3805</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 3: Return Code 08 Default Error Message Assignments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Error Codes</strong></td>
</tr>
<tr>
<td>-----------------</td>
</tr>
<tr>
<td>CLI0530</td>
</tr>
<tr>
<td>2123</td>
</tr>
<tr>
<td>2538</td>
</tr>
<tr>
<td>2541</td>
</tr>
<tr>
<td>2632</td>
</tr>
<tr>
<td>2639</td>
</tr>
<tr>
<td>2641</td>
</tr>
</tbody>
</table>
### Table 3: Return Code 08 Default Error Message Assignments (continued)

<table>
<thead>
<tr>
<th>Error Codes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2644</td>
<td>No more room in database %DBID.</td>
</tr>
<tr>
<td>2654</td>
<td>Operation not allowed: %DBID.%TVMID is being restored.</td>
</tr>
<tr>
<td>2805</td>
<td>Maximum row length exceeded in %TVMID.</td>
</tr>
<tr>
<td>2809</td>
<td>Invalid recovery sequence detected.</td>
</tr>
<tr>
<td>2815</td>
<td>Apparent invalid restart of a restore.</td>
</tr>
<tr>
<td>2818</td>
<td>Invalid lock to dump table without after image journaling.</td>
</tr>
<tr>
<td>2825</td>
<td>No record of the last request was found after Teradata Database restart.</td>
</tr>
<tr>
<td>2826</td>
<td>Request completed but all output was lost due to Teradata Database restart.</td>
</tr>
<tr>
<td>2827</td>
<td>Request was aborted by user or due to statement error.</td>
</tr>
<tr>
<td>2828</td>
<td>Request was rolled back during system recovery.</td>
</tr>
<tr>
<td>2830</td>
<td>Unique secondary index must be dropped before restoring table.</td>
</tr>
<tr>
<td>2835</td>
<td>A unique index has been invalidated. Resubmit request.</td>
</tr>
<tr>
<td>2837</td>
<td>Table being fast loaded; no data dumped.</td>
</tr>
<tr>
<td>2838</td>
<td>Table is unhashed; no data dumped.</td>
</tr>
<tr>
<td>2840</td>
<td>Data rows discarded due to inconsistent hash codes.</td>
</tr>
<tr>
<td>2843</td>
<td>No more room in database.</td>
</tr>
<tr>
<td>2866</td>
<td>Table was recovery aborted; no data dumped.</td>
</tr>
<tr>
<td>2868</td>
<td>This permanent journal table is damaged; no data dumped.</td>
</tr>
<tr>
<td>2920</td>
<td>Delete journal and AMP down without dual.</td>
</tr>
<tr>
<td>2926</td>
<td>No more room in %DBID.%TVMID.</td>
</tr>
<tr>
<td>3001</td>
<td>Session is already logged on.</td>
</tr>
<tr>
<td>3111</td>
<td>The dispatcher has timed out the transaction.</td>
</tr>
<tr>
<td>3116</td>
<td>Response buffer size is insufficient to hold one record.</td>
</tr>
<tr>
<td>3119</td>
<td>Continue request submitted but no response to return.</td>
</tr>
<tr>
<td>3120</td>
<td>The request is aborted because of a Teradata Database recovery.</td>
</tr>
<tr>
<td>3523</td>
<td>%FSTR does not have %VSTR access to %DBID.%TVMID.</td>
</tr>
<tr>
<td>3524</td>
<td>%FSTR does not have %VSTR access to data base %DBID.</td>
</tr>
<tr>
<td>3566</td>
<td>Data base does not have a PERMANENT journal.</td>
</tr>
</tbody>
</table>
### Chapter 3: Using BTEQ

#### Handling Errors

3596 RESTORE Teradata Database invalid if table, view or macro exists outside of Teradata Database.

3598 Concurrent change conflict on database; try again.

3603 Concurrent change conflict on table; try again.

3613 Dump/restore, no hashed nonfallback tables found.

3656 Journal table specified no longer exists.

3658 ROLLBACK/ROLLFORWARD table specifications are invalid.

3705 Teradata SQL request is longer than the Simulator maximum.

3802 Database “%VSTR” does not exist.

3807 Table/view “%VSTR” does not exist.

3824 Macro “%VSTR” does not exist.

3873 “%VSTR” is not a journal table.

3877 NO FALLBACK specified and the table is FALLBACK.

3897 Request aborted due to Teradata Database restart. Resubmit.

3916 Requested information not in dictionary.

5495 Stored Procedure %VSTR does not exist.

### Table 3: Return Code 08 Default Error Message Assignments (continued)

<table>
<thead>
<tr>
<th>Error Codes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLI0001</td>
<td>Parameter list invalid or missing.</td>
</tr>
<tr>
<td>CLI0002</td>
<td>Invalid number of parameters received.</td>
</tr>
<tr>
<td>CLI0003</td>
<td>Error validating HSIRCB.</td>
</tr>
<tr>
<td>CLI0004</td>
<td>Error validating HSICB.</td>
</tr>
<tr>
<td>CLI0005</td>
<td>Error validating HSISPB.</td>
</tr>
<tr>
<td>CLI0006</td>
<td>Invalid destination HSICB detected.</td>
</tr>
<tr>
<td>CLI0007</td>
<td>Invalid destination RCB detected.</td>
</tr>
<tr>
<td>CLI0008</td>
<td>DBCFRC unable to free RCB/HSICB control blocks because they are not contiguous in storage.</td>
</tr>
<tr>
<td>CLI0009</td>
<td>Invalid DBCAREA pointer or id.</td>
</tr>
<tr>
<td>CLI0010</td>
<td>ECB already waiting.</td>
</tr>
</tbody>
</table>
Error Severity Levels

As described in the preceding subsection, the default return code assignments for Teradata Database error messages also determine the internal BTEQ ERRORLEVEL value that you can use in subsequent testing and branching commands, such as:

```
SELECT * FROM SOMEWHERE;
IF ERRORLEVEL >= 14 THEN QUIT 17;
```

You can also use the BTEQ ERRORLEVEL command to change the severity level associated with one or more error codes, as in the following example:

```
SET ERRORLEVEL 2168 SEVERITY 4,
     (2173, 3342, 5262) SEVERITY 8
SET ERRORLEVEL UNKNOWN SEVERITY 16
```

**Note:** As shown, you can also use UNKNOWN as a wildcard error number to accommodate any error codes that are not otherwise defined.

Maximum Error Level

Other than the return code submitted to your client operating system, BTEQ error severity levels have no immediate impact on operations. Unless you incorporate them into your scripts or macros, each task or job continues regardless of the assigned level of error severity. The MAXERROR attribute, however, defines a threshold value that, when encountered, automatically terminates BTEQ job processing.

If a BTEQ script contains a Teradata SQL statement that produces an error severity level that is greater than the designated MAXERROR value, BTEQ immediately aborts that job.

Use the following MAXERROR command, for example, to terminate BTEQ whenever a Teradata Database error that is assigned an error level of 08 or higher occurs:

```
SET MAXERROR 8
```

If you do not specify a MAXERROR value, BTEQ jobs execute until one of the following conditions occurs:

- End-of-file for the primary command input file is encountered.
- A QUIT command is processed.
- A fatal error is detected.
  - When BTEQ receives an I/O abend, system error messages appear in the z/OS JES job log
  - When BTEQ receives an I/O error or an abend, the SAS/C runtime library produces an LSCX message that may provide more information about the error

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2971</td>
<td>The AMP lock table has overflowed.</td>
</tr>
<tr>
<td>2972</td>
<td>No table header exists for table.</td>
</tr>
</tbody>
</table>
For more information on I/O errors and abends, refer to “I/O Errors and Abends” on page 92.

**Stored Procedure Compilation**

The Stored Procedure Language (SPL) compilation errors and warnings are reported by Teradata Database as part of the SUCCESS/OK parcel in response to the COMPILE command.

All syntax and semantic errors in the stored procedure source text are reported with a non-zero warning code. The activity count is set to the total number of compilation errors and warnings.

Teradata Database server cannot report more than 64 KB of SPL compilation errors and warnings. If they exceed the limit, the server returns only the first 64 KB of errors and warnings, and ignores all remaining errors. If the last error or warning text does not fit in the 64-KB buffer, it is eliminated. A message is given that indicates too many compilation errors have occurred.

The stored procedure is not created or replaced if compilation errors are found. Instead, the stored procedure is created or replaced if compilation warnings are found, and no failures are reported.

**Structure**

SPL compilation error or warning messages are uniquely identifiable. The error code begins with “SPL”, followed by a four-digit number.

This number is followed by an “E” that indicates an error, or “W” that indicates a warning. A line number in parenthesis indicates where in the stored procedure source text the error or warning was detected. An appropriate error/warning text follows the code.

**Response Mode**

The organization of the SPL compilation errors and warnings in the response depends on the response mode in which the COMPILE request was submitted.

<table>
<thead>
<tr>
<th>Response Mode</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td>each error or warning is displayed as a row on a separate line. The error or warning text is in a readable format.</td>
</tr>
<tr>
<td>Record, Indicator, or Multipart Indicator</td>
<td>each error or warning is organized as a series of bytes in a data parcel. BTEQ interprets each data parcel and prints the errors and warnings.</td>
</tr>
</tbody>
</table>

**BTEQ Error Output Files**

In response to your BTEQ commands and Teradata SQL statements, BTEQ produces two kinds of output:

- Error messages
- Output data
Normally, by default, BTEQ writes both error messages and output data to the standard STDOUT file. From there, the output is directed to either your terminal or some other device, according to the file definitions that are in effect when you invoke BTEQ.

By using the BTEQ ERROROUT command, you can direct the error messages to some other file or device via the STDERR file. Before doing so, however, you must define the ddname of SYSTERM for the STDERR device in your JCL before logging on to BTEQ, or by using the BTEQ TSO command.

Then, to specify a different output file for error messages, enter the ERROROUT command as follows:

```
.errorout stderr
*** Error messages now directed to STDERR.
```

**I/O Errors and Abends**

I/O errors are handled differently by BTEQ in a channel-attached system than in a network-attached system. In a network-attached system operating in batch mode, BTEQ will report an I/O error to the user, then continues to execute the next SQL or BTEQ command. In a channel-attached system, BTEQ reports the I/O error, then terminates the job with an abend.

The next section explains how the C version of BTEQ handles I/O errors and abends for channel-attached clients.

**Channel-Attached Clients:**

- BTEQ intercepts I/O abends and either converts them to a user 1000 abend or handles the error, allowing the program to continue.
- When BTEQ converts an abend to a user 1000, it provides a reason code in general register 15 that gives additional information on the abend. The reason code associated with the abend is the C error number value.

The following is a list of error number values that cause abends:

- No list exists that explains which system abend (for example, B37, E37, 80A) equates to which reason code. This is a flaw in the SAS/C documentation.
- When BTEQ receives an I/O abend, system error messages appear in the z/OS JES job log.
- When BTEQ receives an I/O error or an abend, the SAS/C runtime library produces an LSCX (C runtime) message that may provide additional information about the error.
- JCL applications that presume a return code of zero may terminate because of nonzero default return codes even if no other errors occur.

**Handling an I/O Abend on z/OS**

BTEQ converts an abend to user 1000. This is particularly useful to automate processes based on job step condition code checking, as shown in the following example:

```c
//YNBTQ JOB 1,'Your Name',CLASS=B,Notify=YN,
// MSGCLASS=A,MSGLEVEL=(1,1)
```
BTEQ Large File (LF) Support (for files >2GB)

In the example, the program CRATDTFL is executed if the attempt to run the BTEQ program results in a user abend 1000 due to an I/O error. If the BTEQ program runs successfully, DOSMTHNG is executed.

### BTEQ Large File (LF) Support (for files >2GB)

The 64-bit BTEQ on HPUX (Itanium ia-64) automatically supports Large Files (LF) for files greater than 2GB. Large File (LF) support has been added for 32-bit BTEQ on all supported UNIX platforms.

There are two exceptions to the 32-bit BTEQ Large Files support.

32-bit BTEQ will fail if:

- the I/O Encoding used is UTF16, and a Large RUN file is being read.
- the Session CharSet is UTF16, and a TEXT (REPORT/VA'RETEXT) IMPORT from a Large IMPORT file is being done.

In both the above cases, BTEQ uses 32-bit Shared ICU Libraries for Teradata (TDICU) functions to handle the respective files. Since the 32-bit TDICU does not support Large Files, BTEQ will fail.
This chapter provides information about handling output from BTEQ:

- Formatting BTEQ Output: Default Specifications
- Converting Screen Displays to Reports
- Resubmitting Output as Input

## Formatting BTEQ Output: Default Specifications

All Teradata sessions within a set operate under the same formatting options, such as screen character width, page length, headers and footers, column spacing, and so on.

### How Defaults Are Inherited

The first set of Teradata sessions in a BTEQ execution automatically adopts the default format specifications.

If you make changes to the format specifications during the first session, then the second set of sessions within the same BTEQ execution inherits the format specifications as they existed at the end of the first set of sessions.

Subsequent sessions do not automatically revert to the default format specifications.

### Viewing the Current Format Specifications

Use the BTEQ SHOW CONTROLS command whenever you want to see the current format specifications:

```
.SHOW CONTROLS
```

See “SHOW CONTROLS” on page 289 for a complete listing of the format specifications display.

### Specifying the Default Format Specifications

The default format specifications are appropriate for interactive/online BTEQ sessions. If you have changed a number of the format specifications to accommodate special reports or BTEQ scripts or macros, you can use the BTEQ DEFAULTS command to return to the default specifications:

```
.SET DEFAULTS
```
Changing the Screen Width

The default WIDTH setting accommodates display of result lines comprised of a maximum of 75 characters (assuming they all have single-byte representations if not unicode) for interactive use. If your terminal screen can display more than this, you may want to increase the WIDTH setting. This may be of particular value if you are entering Teradata SELECT statements with so many expressions that you are unable to see your full results using the default setting. The valid range is from 20 to 65531.

Example

Use the following BTEQ WIDTH command, for example, to change the value of the screen width specification to 120 characters:

```
.SET WIDTH 120
```

If you specify a screen width that is smaller than 20 characters or larger than 65531 characters, BTEQ displays the following error message:

```
*** Error: Width out of legal range of 20 to 65531.
```

For details, see “WIDTH” on page 322.

Changing the Null Specification

The NULL specification identifies the character or a string of characters that BTEQ uses to represent null value fields in the reported results of a Teradata SQL SELECT statement. The BTEQ NULL specification can range from 1 to 253 characters. The value is a single question mark character by default.

Example

Use the following BTEQ NULL command, for example, to specify a string of three hyphens as the null specification:

```
.SET NULL AS '---'
```

For details, see “NULL” on page 229.

Disabling the Echo Required Function

By default, the echo required function (ECHOREQ command) is enabled. This returns an exact copy of every BTEQ command and Teradata SQL statement to the standard output stream. When using BTEQ interactively, this means that BTEQ redisplay (echoes) everything you enter; you will see everything twice.
Example

Use the following BTEQ ECHOREQ command to disable the echo required function:

```sql
.SET ECHOREQ OFF
```

The BTEQ ECHOREQ command is not related to the Teradata SQL ECHO statement.

Disabling the Retry Function

By default, the BTEQ request retry function (RETRY command) is enabled. This automatically resubmits requests that fail because of certain error conditions.

Use the BTEQ RETRY command to disable the retry function before submitting requests that are embedded in either an explicit transaction, a multi-statement request, or an executing BTEQ macro.

Note: For Teradata transaction semantics mode, explicit transactions begin with a BEGIN TRANSACTION statement and end with an END TRANSACTION statement. Multi-statement requests and executing BTEQ macros are treated as implicit transactions.

When an SQL Statement Fails in Teradata Mode

When any statement in a Teradata Mode request fails, BTEQ returns a failure message. When a Teradata SQL statement fails, Teradata Database aborts the entire request and backs out every statement in the request, not just the one that failed. And, if the request uses Teradata semantics rather than ANSI semantics and was embedded in a transaction, Teradata Database aborts the entire transaction and backs out every statement in the transaction. The result is that the database reverts back to the state it was in when you initiated the transaction.

The BTEQ RETRY command, however, only resubmits the request that failed. If the request was embedded in a transaction, or in either a multi-statement request or an executing macro, then the effects of any statements before the failed request are lost, as is any indication that a database transaction is in progress. Resubmitting only the request that failed, in these situations, would most likely introduce undesirable changes to the database.

Example

Setting the RETRY command option to ON does not work for transactions coded like the following example because Teradata Database rolls back the transaction to and including the BEGIN TRANSACTION statement if one of the subsequent statements fails:

```sql
BEGIN TRANSACTION ;
UPDATE ... ;
UPDATE ... ;
END TRANSACTION ;
```

After the rollback, BTEQ retries the statement that failed. If the first UPDATE statement failed, BTEQ retries starting at the first UPDATE statement.
Example

BTEQ resubmits requests, not transactions. If you want to resubmit an entire transaction, write it as a multi-statement single request by employing just one trailing semicolon. This alternate form of script will handle the transaction semantics properly with RETRY ON.

The SQL statements MUST look like this...

```sql
BEGIN TRANSACTION
; UPDATE ...
; UPDATE ...
; END TRANSACTION ;
```

Example

This is an alternate method for coding transactions:

```sql
BT;
UPDATE...;
. IF ERRORCODE <> 0 THEN .GOTO FORGETIT
UPDATE...;
. IF ERRORCODE <> 0 THEN .GOTO FORGETIT
UPDATE...;
. IF ERRORCODE <> 0 THEN .GOTO FORGETIT
ET;
. LABEL FORGETIT
```

Setting the RETRY command option to OFF and testing the error code after each request flushes the remaining statements. Be careful, however, when using this method. If you are executing a macro within a transaction, it must follow the same logic because BTEQ executes each statement from the macro individually. Just testing after the EXEC statement does not work because the test applies only to the last statement in the macro.

Use the following BTEQ RETRY command to disable the retry function:

```sql
.SET RETRY OFF
```

Specifying Record or Indicator Mode

By default, BTEQ initializes in Field Mode to support interactive operations. Field Mode is defined when both the RECORDMODE and INDICDATA command options are set to OFF. There is no corresponding FIELD command to specify the Field Mode.

You can use Record or Indicator Mode in interactive operations, however, when you need to test scripts or macros that send responses to an export file. This lets you see the data that would be diverted to the export file on your terminal screen. A “hex-dump” style output will be generated. Leading asterisks may be placed in the results to indicate the end of a sequential set of result lines that are identical. For example, the asterisk following 2710 in the fourth line below indicates that the lines for 0010 through 2700 are identical.

```plaintext
*** Record#1. Dump of Data:
 0000  0030 3510 2700 0000  0000 00AA BBCC 0000  *.05.'...........*
 0010  0000 0000 0000 0000  0000 0000 0000 0000  *................*
2710 *0000 0000 0000 0000  DDEE FF *...........*
```
Normally, you would specify either Record Mode or Indicator Mode in batch operations when:

- Using the BTEQ EXPORT command to direct output data from Teradata Database to an export file either for reloading later or for transfer to another system. For details, see “EXPORT” on page 152.
- Using the BTEQ IMPORT command to direct input data from a client system file to Teradata Database. For details, see “IMPORT” on page 190.

**Example**

**Note:** Use the following BTEQ RecordMode or IndicData commands to specify Record or Indicator Mode. If the return data will have null values, use the IndicData command; if not, use the RecordMode command.

```
.SET RECORDMODE ON
```
or

```
.SET INDICDATA ON
```

Note that all of the specified data is selected. Unless you need to display all of the return data, use the BTEQ RETCANCEL and RETLIMIT commands, as described in the following subsection, to limit the size of the display.

When you are ready to return to Field Mode and resume normal batch operations, use the appropriate command to discontinue Record or Indicator Mode:

```
.SET RECORDMODE OFF
```
or

```
.SET INDICDATA OFF
```

**Numeric Overflow in Reports**

When in Field Mode, a numeric overflow error returned for numeric or decimal data types in a report, is reported as ‘***’ instead of ‘Error.... etc.’ The following query returns the correct format because the format accommodates the number of digits returned:

```
SELECT cast (123456 as integer format '$999,999');
```

The result is:

```
123456
$123,456
```

But the next query returns an overflow reported as stars ‘***...’:

```
SELECT CAST (123456 as integer format '$99,999');
```

The result is:

```
123456
******
If the same query is executed in Record or Indicator Mode, then the numeric overflow is reported as an error, the preferred response.

## Limiting the Amount of Display Data

When using either Record or Indicator Mode in an interactive session, you may want to limit the number of rows of data displayed on your terminal screen. If, for example, you are developing the format portion of a BTEQ script or macro, you do not need to see all of the data. You need only see a few rows to evaluate the results of your format statements.

Use the following BTEQ commands to limit the number of rows of data in your screen display:

<table>
<thead>
<tr>
<th>Comment</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>RETCANCEL</td>
<td>Use this command to cancel the request after displaying the specified number of rows</td>
</tr>
<tr>
<td>RETLIMIT</td>
<td>Use RETLIMIT’s ROWS option to specify the number of rows of return data you want to display on your terminal screen.</td>
</tr>
</tbody>
</table>

**Note:**
- If you specify a return limit value for rows, but leave the RETCANCEL command option set to OFF, BTEQ stops displaying data at the specified limit, but continues processing the request.
- The value of the RETLIMIT command ROWS option is zero by default, which is no limit. If you set the RETCANCEL command option to ON, but do not specify a return limit value for rows, BTEQ ignores the RETCANCEL command and retrieves and displays all of the request data.
- When BTEQ receives the response in Record Mode or Indicator Mode and determines that you have specified the standard output stream instead of an export file, BTEQ represents the data as an octal or hexadecimal dump with side-by-side EBCDIC translation, one dump for each row of results from the SELECT statement.

### Example

This example limits the display to two rows of data:

```
.Set retlimit 2
.Set recordmode on
.select * from fml.department;
```

and produces the following display:

```
*** Success , Stmt#1, ActivityCount = 5
*** Time was 4.70 seconds.
*** Record#1. Dump Of Data:
  0000  F401 000B 456E 6769  6E65 6572 696E 6741  *....EngineeringA*
  0010  544C 1C27  *TL.*
```
*** Record#2. Dump Of Data:
0000  BC02 0009 4D61 726B  6574 696E 674E 5943  *....MarketingNYC*
0010  2527  *%'*
*** RetLimit exceeded.
*** Ignoring the rest of the output for this statement.

Similarly, the following example:

```sql
.SET retlimit 2
.SET indicdata on
select * from fml.department;
```

produces the following display:

```sql
*** Success , Stmt#1, ActivityCount = 5
*** Time was 0.51 second.

*** Dump of Parcel DATAINFO:
0000  0400 F401 0200 C101  0E00 C501 0#00 F401  *................*
0010  0200 *..*
*** Record#1. Dump Of Data:
0000  00F4 0100 0B45 6E67  696E 6752 6E67 7D29 6E67 6E67  *.....Engineering*
0010  4154 4C1C 27  *ATL.*
*** Record#2. Dump Of Data:
0000  00BC 0200 094D 6172  6B65 7469 6E67 4E59  *.....MarketingNY*  
0010  4325 27  *C.*
```

To return to normal interactive/Field Mode, set either the RECORDMODE or the INDICDATA command option to OFF, and reset the RETLIMIT command ROWS option to zero:

```sql
.SET RECORDMODE OFF
```
or

```sql
.SET INDICDATA OFF
```
followed by:

```sql
.SET RETLIMIT 0
```

# Specifying Report Format

## Vertical Presentation

When your Teradata SQL SELECT statements use an ORDERED BY clause, use the following BTEQ commands to control the vertical presentation of data in your reports:

<table>
<thead>
<tr>
<th>Command</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>SKIPLINE and SKIPDOUBLE</td>
<td>Inserts blank lines.</td>
</tr>
<tr>
<td>SUPPRESS</td>
<td>Deletes duplicated data.</td>
</tr>
<tr>
<td>UNDERLINE</td>
<td>Inserts one line of underline characters.</td>
</tr>
</tbody>
</table>

---

Basic Teradata Query Reference 101
**Horizontal Presentation**

Use the following BTEQ commands to control the horizontal presentation:

<table>
<thead>
<tr>
<th>Command</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOLDLINE</td>
<td>Splits report rows into two or more lines.</td>
</tr>
<tr>
<td>SEPARATOR</td>
<td>Specifies the column separator characters.</td>
</tr>
<tr>
<td>SIDETITLE</td>
<td>Positions sidetitles to the left of their summary column data.</td>
</tr>
</tbody>
</table>

The sections that follow describe briefly the vertical and horizontal format specifications that you can make. For more details on each command, refer to Chapter 5: “BTEQ Commands.”

**Deleting Duplicated Data**

The BTEQ SUPPRESS command deletes duplicated data by replacing all consecutively repeating values with blank characters in your output reports.

**Example**

Without using the SUPPRESS command, for example, the following SELECT statement:

```sql
select deptno, name from employee
where deptno in (100, 300)
order by deptno;
```

produces the following report:

<table>
<thead>
<tr>
<th>DeptNo</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>100</td>
<td>Chin M</td>
</tr>
<tr>
<td>100</td>
<td>Greene W</td>
</tr>
<tr>
<td>100</td>
<td>Moffit H</td>
</tr>
<tr>
<td>100</td>
<td>Peterson J</td>
</tr>
<tr>
<td>300</td>
<td>Russell S</td>
</tr>
<tr>
<td>300</td>
<td>Phan A</td>
</tr>
</tbody>
</table>

Adding a BTEQ SUPPRESS command:

```sql
.SET SUPPRESS ON 1
select deptno, name from employee
where deptno in (100, 300)
order by deptno;
```

produces the following report:

<table>
<thead>
<tr>
<th>DeptNo</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>------</td>
<td>---------</td>
</tr>
<tr>
<td>100</td>
<td>Chin M</td>
</tr>
<tr>
<td></td>
<td>Greene W</td>
</tr>
<tr>
<td></td>
<td>Moffit H</td>
</tr>
<tr>
<td></td>
<td>Peterson J</td>
</tr>
</tbody>
</table>
Eliminating the duplicated department numbers in the first column makes it easier to see the break between Dept. 100 and Dept. 300.

Inserting Blank Lines

Instead of, or in addition to, using the BTEQ SUPPRESS command, you can use the SKIPLINE and SKIPDOUBLE commands to insert one or two blank lines in your report whenever the value of a specified column changes.

Example

Adding a BTEQ SKIPLINE command to the previous example:

```
.SET SKIPLINE ON 1
.SET SUPPRESS ON 1
select deptno, name from employee
where deptno in (100, 300)
order by deptno;
```

produces the following report:

```
DeptNo  Name
------  -----------
100  Chin M
     Greene W
     Moffit H
     Peterson J
300  Russell S
     Phan A
     Leidner P
```

Similarly, using a SKIPDOUBLE command would add two blank lines, and using both SKIPLINE and SKIPDOUBLE would add three.

Inserting a Line of Underline Characters

Use the BTEQ UNDERLINE command to insert a row of underline characters across your report whenever the value of a specified column changes. (Use the UNDERLINE command either in addition to, or instead of, any of the other BTEQ vertical formatting commands: SUPPRESS, SKIPLINE, or SKIPDOUBLE.)

Example

Replacing the SKIPLINE command with an UNDERLINE command in the previous example:

```
.SET UNDERLINE ON 1
.SET SUPPRESS ON 1
```
select deptno, name from employee
where deptno in (100, 300)
order by deptno;

produces the following report:

<table>
<thead>
<tr>
<th>DeptNo</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Chin M</td>
</tr>
<tr>
<td></td>
<td>Greene W</td>
</tr>
<tr>
<td></td>
<td>Moffit H</td>
</tr>
<tr>
<td></td>
<td>Peterson J</td>
</tr>
<tr>
<td>300</td>
<td>Russell S</td>
</tr>
<tr>
<td></td>
<td>Phan A</td>
</tr>
<tr>
<td></td>
<td>Leidner P</td>
</tr>
</tbody>
</table>

Specifying Page Breaks

By default, BTEQ uses all of the space available on each page for report information. Use the BTEQ PAGEBREAK command to force a page break and begin with a new page whenever the value of a specified column changes.

Example

Replacing the UNDERLINE command with a PAGEBREAK command in the previous example:

```
.SET PAGEBREAK ON 1
.SET SUPPRESS ON 1
select deptno, name from employee
where deptno in (100, 300)
order by deptno;
```

produces the following two-page report:

page 1:

<table>
<thead>
<tr>
<th>DeptNo</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Chin M</td>
</tr>
<tr>
<td></td>
<td>Greene W</td>
</tr>
<tr>
<td></td>
<td>Moffit H</td>
</tr>
<tr>
<td></td>
<td>Peterson J</td>
</tr>
</tbody>
</table>

page 2:

<table>
<thead>
<tr>
<th>DeptNo</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>Russell S</td>
</tr>
<tr>
<td></td>
<td>Phan A</td>
</tr>
<tr>
<td></td>
<td>Leidner P</td>
</tr>
</tbody>
</table>
Specifying Column Separations

By default, all columns in a BTEQ report are separated by two blank characters. Use the BTEQ SEPARATOR command to specify a different number of blank characters or another character string to separate columns.

Example

Replacing the PAGEBREAK command in the previous example with a SEPARATOR command specifying a single vertical bar as the separator string, for example:

```sql
.SET SEPARATOR "|"
.SET SUPPRESS ON 1
select deptno, name from employee
where deptno in (100, 300)
order by deptno;
```

produces the following report:

```
DeptNo|Name
-----|-----------
100|Chin M
|Greene W
|Moffit H
|Peterson J
300|Russell S
|Phan A
|Leidner P
```

Splitting Report Lines

Horizontal Presentation

By default, BTEQ presents each row of output data as one row in your reports, with each column presented side by side in a horizontal format.

The following SELECT statement, for example:

```sql
SELECT DeptNo, Name, JobTitle, Salary
FROM Employee WHERE DeptNo = 300;
```

produces the following report:

```
DeptNo  Name  JobTitle  Salary
------  ------------  ------------  ---------
300  Russell S  President  65,000.00
300  Phan A  Vice Pres  55,000.00
300  Leidner P  Secretary  23,000.00
```

Vertical Presentation

You can use the BTEQ FOLDLINE command to break each row of output data into two or more rows in your reports. By using the FOLDLINE ON ALL form of the command, you can
effectively change the format of your reports from horizontal to vertical by reporting each column value on a new line.

Adding a FOLDLINE command to the previous example:

```
.SET FOLDLINE ON ALL
SELECT Name, JobTitle, Sex
FROM Employee WHERE DeptNo = 300;
```

reformats the report as follows:

```
Name
------------
JobTitle
------------
Sex
---
Russell S
President
M
Phan A
Vice Pres
F
Leidner P
Secretary
F
```

In this case, you would probably want to use the BTEQ SIDETITLES command, as described in the following subsection, to reposition the column headings to the left of the rows.

**Specifying Side-Titles**

If you have used the BTEQ FOLDLINE command to change the format of your report from horizontal to vertical, as in the previous example, you can add the SIDETITLES command to reposition the headings to the left side of the rows.

**Example**

Adding a SIDETITLES command to the previous example:

```
.SET FOLDLINE ON ALL
.SET SIDETITLES ON
SELECT Name, JobTitle, Sex
FROM Employee WHERE DeptNo = 300;
```

reformats the report as follows:

```
Name  Russell S
JobTitle  President
Sex  M
Name  Phan A
JobTitle  Vice Pres
Sex  F
Name  Leidner P
JobTitle  Secretary
Sex  F
```
Specifying Headers and Footers

You can specify headers and footers for the top and bottom of each page that can include the following information:

- The report date. This is the date that the print file was created, not the date the report was actually printed.
- The report time. This is the time that the print file was created, not the time the report was actually printed.
- Page number.
- The current value of the data in a certain column of the report. Usually, this would be a column specified by an ORDER BY clause in the SELECT statement.
- A running text string that identifies the report. This is usually a descriptive version of the SELECT statement.

If you do not specify a header (or title) for your report, BTEQ uses a default one-line header/title that includes the report date (left-justified), as much of the Teradata SQL SELECT statement that fits (centered), and the page number (right-justified).

If you do not specify a footer, BTEQ uses the available space at the bottom of the page for additional rows of report data.

BTEQ Commands to Use

To specify a heading, use either the HEADING or RTITLE (report title) command. The two commands are variations of the same command. The RTITLE command is easier to use because it automatically provides the report date and page numbers.

To specify a footing, use the FOOTING command.

The command formats are:

```
.SET HEADING 'heading specifier'
.SET RTITLE 'heading specifier'
.SET FOOTING 'footing specifier'
```

The specifier has the following format for both headers and footers:

```
'a||b||c//d||e||f//g||h||i'
```

where:

- `a` is the left-justified entry on line 1.
- `b` is the centered entry on line 1.
- `c` is the right-justified entry on line 1.
- `d` is the left-justified entry on line 2.
- `e` is the centered entry on line 2.
- `f` is the right-justified entry on line 2.
- `g` is the left-justified entry on line 3.
- `h` is the centered entry on line 3.
Specifying Headers and Footers

• \( i \) is the right-justified entry on line 3.

and:

• \( || \) separates the three parts of a line.

• \(/\) separates the lines.

You can break a header line and begin a new line of header text by inserting a pair of slashes (\(/\)/) at the desired break point. BTEQ allows up to nine breaks (10 header lines maximum) within a HEADING or FOOTING command.

You need only to specify the entries that you want to use, with the appropriate separator characters. If you do not specify left-middle-right entries, BTEQ centers the text. If you do not specify first-second-third lines, BTEQ puts your text on one line (the first).

If you specify more than two pair of line separator characters, indicating more than three lines, the following occurs:

<table>
<thead>
<tr>
<th>IF you are using a...</th>
<th>Then BTEQ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>channel-attached system</td>
<td>returns an error message and treats the additional separators as header or footer text.</td>
</tr>
<tr>
<td>network-attached system</td>
<td>truncates the additional separators from the header or footer text.</td>
</tr>
</tbody>
</table>

**Example**

In a channel-attached system, using the following FOOTING command and SELECT statement:

```
.footing '&DATE||Confid||dential||Page&PAGE
select * from department;
```

produces the following error message and report:

```
.footing '&DATE||Confid||dential||Page&PAGE
*** Warning: Maximum of 2 '||' separators is allowed, others ignored.
*** Query completed. 5 rows found. 4 columns returned.
90/08/10 select * from department; Page 1
```

<table>
<thead>
<tr>
<th>DeptNo</th>
<th>DeptName</th>
<th>Loc</th>
<th>MgrNo</th>
</tr>
</thead>
<tbody>
<tr>
<td>------</td>
<td>---------------</td>
<td>----</td>
<td>------</td>
</tr>
<tr>
<td>500</td>
<td>Engineering</td>
<td>ATL</td>
<td>10012</td>
</tr>
<tr>
<td>700</td>
<td>Marketing</td>
<td>NYC</td>
<td>10021</td>
</tr>
<tr>
<td>300</td>
<td>Exec Office</td>
<td>NYC</td>
<td>10018</td>
</tr>
<tr>
<td>600</td>
<td>Manufacturing</td>
<td>CHI</td>
<td>10007</td>
</tr>
<tr>
<td>100</td>
<td>Administration</td>
<td>NYC</td>
<td>10011</td>
</tr>
</tbody>
</table>

Refer to “HEADING” on page 181, “RTITLE” on page 267, and “FOOTING” on page 169 for additional information and examples of specifying report headers and footers.
Specifying Printer Formats

Use the following BTEQ commands to specify the printer formats:

- WIDTH – Specifies the number of characters per line.
- PAGELENGTH – Specifies the number of lines per page.
- FORMCHAR – Specifies the form-feed character.

Though you can change the printer format specifications at any time, you must set BTEQ FORMAT command option to ON for them to take effect.

By default, the BTEQ FORMAT command option is set to OFF. In this condition, BTEQ disregards all of the printer format specifications (except WIDTH), and automatically centers all reports based on the WIDTH specification.

Always set the FORMAT command option to ON when you want to print reports using the printer format specifications.

Specifying the Page Size

Use the BTEQ WIDTH and PAGELENGTH commands to indicate the size of the paper for your report.

Standard 8-1/2 x11 inch and 11x14 inch fan-fold computer paper accommodates the following page size specifications:

- 8-1/2 x11 inch – Width = 80 characters; length = 66 lines
- 11x14 inch – Width = 123 characters; length = 66 lines

If you are using special nonstandard printer forms, you have to measure them to determine the appropriate print width and length specifications.

Example

To specify the width of the paper, enter:

```
.SET WIDTH n
```

where \( n \) is the number of print characters. (The value is 75 by default, which is appropriate for a typical display screen.)

To specify the length of the paper, enter:

```
.SET PAGELENGTH n
```

where \( n \) is the number of lines per page. (The value is 55 by default, which provides a print gap of 11 lines at the form perforations, with several lines for headers and footers.)
Specifying the Form Feed Character

If your printer requires a special numeric sequence in hexadecimal format to produce a form feed at the end of a page, use the BTEQ FORMCHAR command to specify the sequence.

WITH Clause Output

A SELECT SQL statement that has a WITH clause specifies summary lines and breaks (grouping conditions) that determine how selected results are returned (typically for subtotals). However, the output can vary, depending on:

- the BTEQ commands used
- the order of fields involved in the SELECT and WITH clauses
- result line space availability for summary titles

Sequence of Fields

When columns in WITH clauses are referred to in the same order as they are listed in their associated SELECT clause, the return order for summary values enables BTEQ to position the summary text for all the aggregate operations in the same single result line. If the ordering does not match, BTEQ will create multiple result lines for the summary text. A separate result line will be created for each aggregate operation value.

Example

The next example is a simple SELECT WITH statement in which the sequence of WITH clause column references matches the sequence of SELECT clause column references. BTEQ positions the resulting summary text within the same single result line.

```
SELECT salary, edlev, hcap
FROM employee
WITH
    sum(salary), sum(edlev), sum(hcap)
WHERE deptno = 500;
```

BTEQ Response

*** Query completed. 8 rows found. 3 columns returned.
*** Total elapsed time was 1 second.

<table>
<thead>
<tr>
<th>Salary</th>
<th>EdLev</th>
<th>HCap</th>
</tr>
</thead>
<tbody>
<tr>
<td>34,000.00</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>44,000.00</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>42,000.00</td>
<td>18</td>
<td>0</td>
</tr>
<tr>
<td>30,000.00</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>40,000.00</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>56,000.00</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>22,000.00</td>
<td>18</td>
<td>0</td>
</tr>
</tbody>
</table>
All summary titles are lost because BTEQ cannot find space to fit their text into the same single result line.

**Example**

The next example is a simple SELECT WITH statement in which the sequence of WITH clause column references does not match the sequence of SELECT clause column references. BTEQ positions the resulting summary text within multiple result lines.

```sql
SELECT salary, edlev, hcap
FROM employee
WITH sum(hcap), sum(edlev), sum(salary)
WHERE deptno = 500;
```

**BTEQ Response**

```sql
*** Query completed. 8 rows found. 3 columns returned.
*** Total elapsed time was 1 second.

<table>
<thead>
<tr>
<th>Salary</th>
<th>EdLev</th>
<th>HCap</th>
</tr>
</thead>
<tbody>
<tr>
<td>44,000.00</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>34,000.00</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>42,000.00</td>
<td>18</td>
<td>0</td>
</tr>
<tr>
<td>30,000.00</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>40,000.00</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>56,000.00</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>22,000.00</td>
<td>18</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sum(HCap)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

268,000.00

Summary titles have been included as part of the summary text results, with the exception of the first aggregate value. The first title remains lost due to lack of result line space.

**Summary Title Accommodation**

In the previous example, BTEQ was unable to accommodate the display of the first summary title because no room existed for it. Increasing the first column's width will not alleviate this problem. BTEQ will not place a summary title in result line space that is designated for a summary text aggregate value. To alleviate this constraint, select empty column(s) that are aligned to provide dedicated result line space. The next example shows before and after results using this technique.

**Example**

Assume a table called FruitTrees exists in your current database and that you want to produce a report of how many trees are ripe, plus the related yields. Your table has these columns and values:
Your script contains these BTEQ commands and SQL statements, which do not specify selection of any empty columns:

```
.SET SUPPRESS ON 1
SELECT   Status
 , Fruit
 , Yield
 , TreeId
FROM     FruitTrees
GROUP BY Status, Fruit, Yield, TreeId
ORDER BY Status, Fruit, Yield
WITH     COUNT(Fruit)   (TITLE 'Trees'    )
 , SUM  (Yield)   (TITLE 'Yield'    ) BY Status, Fruit
WITH     COUNT(Status)  (TITLE 'Tot Trees')
 , SUM  (Yield)   (TITLE 'Tot Yield') BY Status
;
```

**BTEQ Response**

```
Status       Fruit     Yield  TreeId
-----------  --------  -----------  ----------
NOT RIPE     APPLE     80    POME02
             APPLE     100   POME01
Trees        2          180
             ORANGE    10    CITR01
             ORANGE    30    CITR02
Trees        2          40
-----------               -----------
4    Tot Yield          220
RIPE         APPLE     20    POME04
             APPLE     400   POME03
Trees        2          420
             ORANGE    500   CITR03
Trees        1          500
-----------               -----------
3    Tot Yield          920
```
Example

In the prior example, BTEQ only finds space for two of the four summary text titles. If the script is changed, as shown in the next example, to include selection of empty columns that create result space for the titles, BTEQ is able to include them. Note that the selected width for the empty columns is important. The titles will be truncated if the width is too small.

```sql
.SET SUPPRESS ON 1,2
SELECT   ' ' (CHAR(9)) (TITLE ''), Status,
         ' ' (CHAR(5)) (TITLE ''), Fruit,
         ' ' (CHAR(5)) (TITLE ''), Yield,
         , TreeId
FROM     FruitTrees
GROUP BY Status, Fruit, Yield, TreeId
ORDER BY Status, Fruit, Yield
WITH     COUNT(Fruit)   (TITLE 'Trees' )
         , SUM  (Yield)   (TITLE 'Yield' ) BY Status, Fruit
WITH     COUNT(Status)  (TITLE 'Tot Trees')
         , SUM  (Yield)   (TITLE 'Tot Yield') BY Status
;
```

**BTEQ Response**

<table>
<thead>
<tr>
<th>Status</th>
<th>Fruit</th>
<th>Yield</th>
<th>TreeId</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOT RIPE</td>
<td>APPLE</td>
<td>80</td>
<td>POME02</td>
</tr>
<tr>
<td></td>
<td>APPLE</td>
<td>100</td>
<td>POME01</td>
</tr>
<tr>
<td></td>
<td>ORANGE</td>
<td>10</td>
<td>CITR01</td>
</tr>
<tr>
<td></td>
<td>ORANGE</td>
<td>30</td>
<td>CITR02</td>
</tr>
<tr>
<td></td>
<td>Trees</td>
<td>2</td>
<td>Yield</td>
</tr>
<tr>
<td></td>
<td></td>
<td>180</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ORANGE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>CITR01</td>
</tr>
<tr>
<td></td>
<td>ORANGE</td>
<td>30</td>
<td>CITR02</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>Yield</td>
</tr>
<tr>
<td></td>
<td></td>
<td>40</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tot</td>
<td>Yield</td>
</tr>
<tr>
<td></td>
<td></td>
<td>220</td>
<td></td>
</tr>
<tr>
<td>Ripe</td>
<td>APPLE</td>
<td>20</td>
<td>POME04</td>
</tr>
<tr>
<td></td>
<td>APPLE</td>
<td>400</td>
<td>POME03</td>
</tr>
<tr>
<td></td>
<td>ORANGE</td>
<td>500</td>
<td>CITR03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>Yield</td>
</tr>
<tr>
<td></td>
<td></td>
<td>500</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tot</td>
<td>Yield</td>
</tr>
<tr>
<td></td>
<td></td>
<td>920</td>
<td></td>
</tr>
</tbody>
</table>

**Title Phrases**

By default, when using Field Mode for response generation, BTEQ attempts to include the summary text titles as returned by the database. The actual text returned depends on whether defaults are used and/or specific text is indicated through the WITH clause TITLE phrases.

In addition to specifying non-default title text, you can use TITLE phrases to completely eliminate specific summary text titles from BTEQ results. You can do this by specifying a title
composed solely of blank characters. Note, if you specify a null title (with no characters at all), the display of the summary text title dashes will be turned off as well as the title phrase.

**Example**

The next SELECT statement and response shows the technique of using titles comprised solely of blank characters to suppress summary title text generation in results.

```
SELECT
empno, name, salary
FROM
employee
WITH
  sum(salary)(title ' ')
WHERE
  DeptNo = 500;
```

**BTEQ Response**

*** Query completed. 8 rows found. 3 columns returned.  
*** Total elapsed time was 1 second.

<table>
<thead>
<tr>
<th>EmpNo</th>
<th>Name</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>10014</td>
<td>Inglis C</td>
<td>34,000.00</td>
</tr>
<tr>
<td>10016</td>
<td>Carter J</td>
<td>44,000.00</td>
</tr>
<tr>
<td>10004</td>
<td>Smith T</td>
<td>42,000.00</td>
</tr>
<tr>
<td>10010</td>
<td>Reed C</td>
<td>30,000.00</td>
</tr>
<tr>
<td>10012</td>
<td>Watson L</td>
<td>56,000.00</td>
</tr>
<tr>
<td>10015</td>
<td>Omura H</td>
<td>40,000.00</td>
</tr>
<tr>
<td>10009</td>
<td>Marston A</td>
<td>22,000.00</td>
</tr>
</tbody>
</table>

----------

268,000.00

**Example**

The following SELECT statement and response shows the technique of using null titles to suppress summary title dashes as well as summary title text generation in results.

```
SELECT
empno, name, salary
FROM
employee
WITH
  sum(salary)(title '')
WHERE
  DeptNo = 500;
```

**BTEQ Response**

*** Query completed. 8 rows found. 3 columns returned.  
*** Total elapsed time was 1 second.

<table>
<thead>
<tr>
<th>EmpNo</th>
<th>Name</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>10014</td>
<td>Inglis C</td>
<td>34,000.00</td>
</tr>
<tr>
<td>10016</td>
<td>Carter J</td>
<td>44,000.00</td>
</tr>
<tr>
<td>10004</td>
<td>Smith T</td>
<td>42,000.00</td>
</tr>
<tr>
<td>10010</td>
<td>Reed C</td>
<td>30,000.00</td>
</tr>
<tr>
<td>10015</td>
<td>Omura H</td>
<td>40,000.00</td>
</tr>
<tr>
<td>10012</td>
<td>Watson L</td>
<td>56,000.00</td>
</tr>
</tbody>
</table>

268,000.00
Converting Screen Displays to Reports

You can convert screen displays to reports when using BTEQ interactively or with a run file. In addition, on network-attached systems, you can do the same with a standard input file.

Example

The following input file displays on the screen:

```
.logon tdpid/userid, password
execute deptdisplay;
.logoff
```

Producing a Report

To produce a report, make these changes to the file:

1. Separate the responses to the SQL SELECT statement from the stream of requests and commands that precede and follow it.
2. Format the report with respect to a page to skip the perforations on computer paper.
3. Add page headings with dates and page numbers, centered horizontally on the page.

Changing the Command Sequence

You need to make only a few changes to the sequence of requests and commands to produce a report. Use your editor to add the following commands to the file.

1. Insert an `EXPORT` command after the `LOGON` command to divert the report results to a file called `report.out`.
   
   For channel-attached systems, enter:
   ```
   .tso allocate ddname(repfile) dsname(report.out) shr
   .export report ddname=repfile
   ```
   
   For network-attached systems, enter:
   ```
   .export report file=report.out
   ```

2. After the EXPORT command, enter the following `RTITLE` command to place a report title at the top of each page of the report, along with the date and page number.
   ```
   .set rtitle 'Department Information'
   ```

3. To enable (activate) page formatting commands, add a `FORMAT` command before and after the SQL EXECUTE command as follows:
Producing a Report

4. Use the RESET option of the EXPORT command to end diversion of the report results:

```
.export reset
```

---

**Channel-Attached Example**

This is an example of a file on a channel-attached system:

```
.logon tdpid/userid, password
.tso allocate ddname(repfile) dsname(report.out) shr
.export report ddname=repfile
.set rtitle 'Department information'
.set format on
execute deptdisplay;
.set format off
.export reset
.logoff
```

---

**Network-Attached Example**

This is an example of a file on a network-attached system:

```
.logon tdpid/userid, password
.export report file=report.out
.set rtitle 'Department Information'
.set format on
execute deptdisplay;
.set format off
.export reset
.logoff
.exit
```

---

**Identifying the Input File**

For network-attached systems, invoke BTEQ and specify sampfile as the input file, as follows:

```
BTEQ < sampfile
```

---

**Creating an Output Log File**

To create a new output file named `log.out`, enter:

```
BTEQ < sampfile >log.out
```

The file `log.out` contains:

```
.logon e/fml,
*** Logon successfully completed.
*** Total elapsed time was 3 seconds.
.export report file=report.out

*** To reset export, type .EXPORT RESET
.set rtitle 'Department Information'
.set format on
```
execute deptdisplay;
  *** Echo accepted.
  *** Total elapsed time was 3 seconds.
  *** Query completed. 5 rows found. 4 columns returned.

.set format off
.export reset
  *** Output returned to console.

.logoff
  *** You are now logged off from the DBC.
.exit

Printing the Report File

After examining the files, you can print the report file, and discard the log file:

<table>
<thead>
<tr>
<th>Date</th>
<th>Department Information</th>
<th>Page 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>88/03/07</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeptNo</td>
<td>DeptName</td>
<td>Loc</td>
</tr>
<tr>
<td>------</td>
<td>---------</td>
<td>---</td>
</tr>
<tr>
<td>500</td>
<td>Engineering</td>
<td>ATL</td>
</tr>
<tr>
<td>700</td>
<td>Marketing</td>
<td>NYC</td>
</tr>
<tr>
<td>300</td>
<td>Exec Office</td>
<td>NYC</td>
</tr>
<tr>
<td>600</td>
<td>Manufacturing</td>
<td>CHI</td>
</tr>
<tr>
<td>100</td>
<td>Administration</td>
<td>NYC</td>
</tr>
</tbody>
</table>

Resubmitting Output as Input

You can resubmit output from Teradata Database as input. To do so, you must match the format parameters when you specify your export and import files by specifying the following options for both export and import files:

<table>
<thead>
<tr>
<th>IF your system is...</th>
<th>Then specify...</th>
</tr>
</thead>
<tbody>
<tr>
<td>channel-attached</td>
<td>DATA or INDICDATA.</td>
</tr>
<tr>
<td>network-attached</td>
<td>DATA, INDICDATA or REPORT.</td>
</tr>
</tbody>
</table>

When resubmitting data types, always resubmit fixed-length column data first, followed by variable-length column data.

You cannot reimport data that was exported from Teradata Database in DIF format.

Procedure For Resubmitting Output as Input

To resubmit output as input:

1 Use the BTEQ EXPORT command to specify the export file.
2 Use SQL SELECT statements to retrieve data from Teradata Database.
3 Use the RESET option of the BTEQ EXPORT command to close the export file.
4 Use the BTEQ IMPORT command to specify the file as an import file, using the same format specification that you used for the EXPORT command in step 1.

5 Use SQL statements with USING modifiers to resubmit the output data as input.

**Specifying the Export File**

Use the BTEQ EXPORT command to specify the export file. Be sure to specify a format for the export file that is compatible with the subsequent IMPORT command. The EXPORT command automatically sets the BTEQ RECORDMODE command options to the correct values.

**Specifying REPORT Format**

If you specify REPORT format for your EXPORT command, set the BTEQ FORMAT command option to OFF before entering your SQL SELECT statements. This makes it easier for you to manually reformat the file later.

**Caution:** In REPORT format, BTEQ uses Field Mode to return all data values, including numeric data, in character format. Undesirable results may be obtained with numeric data fields when this data is re-imported.

**Retrieving Data from Teradata Database**

Use SQL SELECT statements to retrieve data from Teradata Database. Each Teradata SQL SELECT statement retrieves information for one set of results. Use the BTEQ REPEAT or = command if you are exporting more than a few records from Teradata Database.

**Retrieving Data in REPORT Format**

If you request data in REPORT format, BTEQ:

- Retrieves both result values and result labels (column titles, etc.)
- Organizes the information into a table
- Writes out the table to the export file, including all labels, using one record (row) for each line.

**Closing the Export File**

Use the RESET option of the BTEQ EXPORT command to close the export file.

If you specified the REPORT format for your export file, it may be helpful to use your system editor to delete all page headers and footers and column heading from the file.

**Note:** Editing your file at this point is helpful, but not necessary. You could skip the format statement when you import the file back to Teradata Database.

**Specifying the Import File**

Use the BTEQ IMPORT command to specify the file as an import file, using the same format specification that you used for the EXPORT command in step 1.
Resubmitting the Output Data
Use SQL statements with USING modifiers to resubmit the output data as input. Use the BTEQ REPEAT or = command if you are importing more than a few records to Teradata Database.

Printing Output on z/OS

The printed output of z/OS mainframes (not .EXPORT DATA or .EXPORT INDICDATA) contains automatically converted characters.

Automatically Converted Characters

The following EBCDIC characters are converted to blanks during printing:

<table>
<thead>
<tr>
<th>Character</th>
<th>Hexadecimal Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>BS: Backspace</td>
<td>X'16'</td>
</tr>
<tr>
<td>CR: Carriage return</td>
<td>X'0D'</td>
</tr>
<tr>
<td>FF: Form feed</td>
<td>X'0C'</td>
</tr>
<tr>
<td>NL: New line</td>
<td>X'15'</td>
</tr>
<tr>
<td>VT: Vertical tab</td>
<td>X'0B'</td>
</tr>
</tbody>
</table>
Chapter 4: BTEQ Output
Printing Output on z/OS
This chapter provides:

- a syntax diagram,
- a description, and
- programming examples

of each BTEQ command. Commands are given in alphabetical order.

For clarity, the repetitive BTEQ prompts and individual command line entries are not shown in the BTEQ responses to programming examples.

Note: Experienced BTEQ users can also refer to the simplified command descriptions in the BTEQ chapter of *Teradata Tools and Utilities Command Summary*. This book provides the syntax diagrams and a brief description of the syntax variables for each Teradata client utility.

### BTEQ Command Syntax

This section describes the correct syntax for BTEQ commands. Using correct syntax requires that you understand the elements of the BTEQ command language and the rules that govern their use.

#### General Rules

Specify BTEQ commands in any combination of uppercase, lowercase, and mixed-case formats. Specify BTEQ commands on a single line, or on multiple lines using the continuation character.

You cannot specify multiple BTEQ commands on a single line, or use a BTEQ command and a Teradata SQL request on the same line.

Signed numbers are not recognized for BTEQ command arguments. All numeric arguments are assumed to be positive. Use of signed numbers may cause unpredictable parsing results. This restriction does not apply to SQL statements submitted through BTEQ.

**Windows OS only:** Enclose all file name variables within single or double quotation marks. In the following example, 'myscript.txt' is the file name:

```
.RUN FILE 'myscript.txt'
```

Additionally, any time text follows a file name, that file name must be enclosed in quotation marks. For example:
.RUN FILE 'myscript.txt' SKIP 2

**BTEQ Command Syntax Example**

The following is an example of the BTEQ FORMAT command showing the correct use of the command elements. FORMAT specifies the conditional use of additional BTEQ output formatting commands.

```
.SET FORMAT ON;
```

In this example, the FORMAT command is set to the ON option, enabling the processing of additional output formatting commands.

**BTEQ Command Elements**

BTEQ commands consist of several elements, some of which are required and some of which are optional. Table 5 describes the elements of BTEQ command syntax, using the previous FORMAT command example.

<table>
<thead>
<tr>
<th>Command Element</th>
<th>Syntax Example</th>
<th>Required or Optional</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>BTEQ command identifier</td>
<td>. (dot character)</td>
<td>Optional, but recommended</td>
<td>Identifies the beginning of a BTEQ command and distinguishes it from Teradata SQL</td>
</tr>
<tr>
<td><strong>Keyword</strong></td>
<td>SET</td>
<td>Optional for many commands</td>
<td>Identifies that the command sets a processing condition</td>
</tr>
<tr>
<td>Command keyword</td>
<td>FORMAT</td>
<td>Required</td>
<td>Specifies the function.</td>
</tr>
<tr>
<td>Options</td>
<td>ON</td>
<td>Required for some commands; otherwise optional</td>
<td>Specifies processing condition.</td>
</tr>
<tr>
<td>Variable</td>
<td>text</td>
<td>Required for some commands; otherwise optional</td>
<td>Specifies a value or entity for processing.</td>
</tr>
<tr>
<td>BTEQ command terminator</td>
<td>; (Semicolon character)</td>
<td>Required</td>
<td>Identifies the end of the BTEQ command.</td>
</tr>
</tbody>
</table>

**Identifying BTEQ Commands**

The initial dot character identifies a statement as a BTEQ command, and distinguishes it from a Teradata SQL request. The dot character enables BTEQ to recognize and execute the command without searching its internal command list, thereby speeding up processing.
Chapter 5: BTEQ Commands

BTEQ Command Syntax

Note: There are no identical BTEQ and Teradata SQL commands.

Continuing BTEQ Commands

You can continue a BTEQ command from one line to the next by including a dash character (-) as the last character of the line to be continued (blank spaces can follow the dash character). For example, the following command:

```
.SET HEADING 'This is an -
extample of a continued -heading'
```

generates the heading:

```
This is an example of a continued heading
```

The continuation character cannot be used in conjunction with an ANSI-style comment statement.

Syntax Variable Definitions

The following variables are used within syntax diagrams.

Table 6: Syntax Variable Definitions

<table>
<thead>
<tr>
<th>Variable</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>bteq_command</td>
<td>BTEQ command</td>
</tr>
<tr>
<td>charstring</td>
<td>character set name</td>
</tr>
<tr>
<td>code</td>
<td>numeric code of character set</td>
</tr>
<tr>
<td>ddname</td>
<td>name of the z/OS JCL DD statement</td>
</tr>
<tr>
<td>errno</td>
<td>error number</td>
</tr>
<tr>
<td>filename</td>
<td>name of network-attached system file</td>
</tr>
</tbody>
</table>

Note: Enclose all file name variables within single or double quotation marks. In the following example, "myscript.txt" is the file name:

```
.RUN FILE "myscript.txt"
```

Also, any time text follows a file name, that file name must be enclosed in quotation marks. For example:

```
.RUN FILE "myscript.txt" SKIP 2
```
Chapter 5: BTEQ Commands

Default BTEQ Values

Table 6: Syntax Variable Definitions (continued)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>'init-string'</td>
<td>name of the initiation string used to start the import process</td>
</tr>
<tr>
<td>modname</td>
<td>name of the file that is to receive the imported data</td>
</tr>
<tr>
<td>operator</td>
<td>comparison operator</td>
</tr>
<tr>
<td>options</td>
<td>z/OS QUEUE option</td>
</tr>
<tr>
<td>os_command</td>
<td>legal command in the current operating system</td>
</tr>
<tr>
<td>sql_request</td>
<td>SQL statement</td>
</tr>
<tr>
<td>string</td>
<td>text string</td>
</tr>
<tr>
<td>text</td>
<td>16-character job identifier</td>
</tr>
<tr>
<td>withlist</td>
<td>list of WITH clause identifiers</td>
</tr>
<tr>
<td>xxx</td>
<td>name of z/OS DD statement</td>
</tr>
</tbody>
</table>

Termination (Semicolon) Character

You must specify the end of a BTEQ command by using the semicolon as the final character in the command syntax. If a BTEQ command is used without the semicolon, BTEQ waits for more input and does not submit the command.

Note: If BTEQ or Teradata Database does not seem to be responding to a command, check for the ending semicolon. If it is missing, type it and press the Enter or Return key.

Terminating a Wait Condition Caused by an Input Error

If you make a typing mistake or forget to close a quote string with a quotation mark, the semicolon that you used as the end-request delimiter is regarded by BTEQ as embedded in the quote string. Consequently, BTEQ waits for more input.

In this case, type the missing quotation mark, then type another semicolon and press Enter or Return. BTEQ might reject the command or request because of the extra semicolon within the quote string, but at least you will have terminated the wait for a missing character.

Default BTEQ Values

Some BTEQ command options are preset to a specific values by default. Within syntax diagrams, default values are listed first among the options, and are represented in underlined text.

Specifying Default Values

When specifying the default option for a command within a script or a macro, always specify the default value explicitly. Specifying command options explicitly makes it easier to
determine exactly what the script or macro does if a problem arises, or if you need to edit or modify.

If the defaults for a command change, having the options specified explicitly will also relieve you of the task of updating scripts or macros that use the command.

**Checking Default Values**

Use SHOW CONTROLS to check the default settings for BTEQ formatting commands.

**BTEQ Command Descriptions**

The following sections describe each BTEQ command:

- The command function
- A syntax diagram
- An explanation of the command options
- Notes on command usage
- Programming examples
### Purpose
Repeats the previous Teradata SQL request a specified number of times.

### Syntax

```
= [n]
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>n</code></td>
<td>the number of times that you want to repeat the previous Teradata SQL request. The valid range is 0 to 2147483647. The default value is 1. If <code>n</code> is 0 (or not specified in the command), BTEQ assumes 1.</td>
</tr>
</tbody>
</table>

### Usage Notes
Use `=` to execute the previous Teradata SQL request one or more additional times. This command enables you to specify a Teradata SQL request, view the results, change formatting options, and execute the request again without having to reenter it.

The definition of `request` is critical to this command. For instance, the following sequence causes only the ET (END TRANSACTION) statement to be submitted 10 more times because BTEQ interprets the ET statement as the last complete request:

```
BT;
select ... ;
select ... ;
select ... ;
ET;
= 10
```

If you enter this by mistake, BTEQ ignores nine ET statements and issues the following message:

"Failure 3510 Too many END TRANSACTION statements."

To submit the entire transaction ten more times, place the semicolon characters as follows:

```
BT
;select ...
;select ...
;select ...
;ET;
```
In this case, because of the placement of the semicolons, BTEQ interprets all of the statements in the transaction as one multi-statement request.

**Difference Between = and REPEAT**

The = command is similar to the REPEAT command with the following differences:

- You must enter REPEAT *before* the request that you want to repeat, and = *after* the request.
- REPEAT specifies the *total* number of times that you want to submit the *following* request; = specifies how many *more* times you want to resubmit the *last* request.
- REPEAT can resubmit a request repetitively until the import file is exhausted; = cannot.

The = command applies to the most recent Teradata SQL request, even if one or more BTEQ commands have occurred after the last Teradata SQL request. Note that this command is most appropriate for the SELECT statement, because it does not actually affect the contents of the database. Always be careful when using the = command to resubmit a statement that affects the contents of the database.

You *cannot* use the = command in a Teradata SQL macro.

**Example 1**

The following example returns the selected information first with the default format, then uses = to repeat the request once after setting the SIDETITLES command option to ON, and twice after setting the FORMAT command option to ON:

```sql
database Personnel;
.def
.format on
select Name DeptNo from Employee
order by Name ;
.def
.format on
.heading ' = 1'
.sidetitles on
= 1
.def
.format on
.heading ' = 2'
= 2
.logoff
.exit
```

**BTEQ Response**

```
... Page 1
Name         DeptNo
------------- -----
Aguilar J    600
Brangle B    700
Carter J     500
Chin M       100
Clements D   700
Greene W     100
Inglis C     500
```
<table>
<thead>
<tr>
<th>Name</th>
<th>DeptNo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aguilar J</td>
<td>600</td>
</tr>
<tr>
<td>Brangle B</td>
<td>700</td>
</tr>
<tr>
<td>Carter J</td>
<td>500</td>
</tr>
<tr>
<td>Chin M</td>
<td>100</td>
</tr>
<tr>
<td>Clements D</td>
<td>700</td>
</tr>
<tr>
<td>Greene W</td>
<td>100</td>
</tr>
<tr>
<td>Inglis C</td>
<td>500</td>
</tr>
<tr>
<td>Kemper R</td>
<td>600</td>
</tr>
<tr>
<td>Leidner P</td>
<td>300</td>
</tr>
<tr>
<td>Marston A</td>
<td>500</td>
</tr>
<tr>
<td>Moffit H</td>
<td>100</td>
</tr>
</tbody>
</table>

(etc)
Example 2
To repeat the previous Teradata SQL request once, type:

=
Chapter 5: BTEQ Commands

ABORT

Purpose
From channel-attached systems, aborts the previous request and returns control to the BTEQ prompt.
Not valid from network-attached client systems.

Syntax

```
ABORT
```

Note: Before you can abort a previous request, you must first send an interrupt to BTEQ. See “Using the Break Key on z/OS” on page 82 for instructions.

Usage Notes

<table>
<thead>
<tr>
<th>When Using the ABORT Command</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>while waiting for an outstanding request,</td>
<td>If the ABORT command is specified while BTEQ is running more than one session, BTEQ initiates the abort procedure on the most recent request on all sessions. Then, if you use the CONTINUE keyword, BTEQ recovers from the interrupt and continues as if the interrupt had never occurred.</td>
</tr>
</tbody>
</table>

| before a request response is sent back, | if the attempted abort reaches Teradata Database before the response to the original request has been sent: |
| | • You receive a failure message because the abort causes the original request to fail. |
| | • The response to the original request is discarded, |
| | • And the transaction in which the original request was embedded, if any, is also aborted. |
| | In this case, the ABORT command backs out any changes that the previous request made to the database. |

| after a request response is sent back, | if the attempted abort reaches Teradata Database after the response to the original request has been sent, the attempted abort is ignored and the original request takes effect. |

| to undo the result of the original request, if it was not embedded, | if the original request was not embedded in a transaction, you can undo the result of the original request by submitting a new request that negates it. |
| | For example, if the original request inserted a row, a new request can delete that row. If the original request was embedded in a transaction, you can undo the results of the original request by submitting a new request consisting of an ABORT command. All requests in the transaction will be aborted, including the original request, and then you can correctly resubmit the transaction. |
**AUTOKEYRETRIEVE**

**Purpose**

Enables users to specify whether the values of any fields associated with Identity Data are returned in response to a SQL Insert operation. Unlike other session control commands, CLI’s SPB files (HSHSPB or clispb.dat) do not provide a means for specifying use of auto generated key retrieval. It can only be invoked at the BTEQ script level through AUTOKEYRETRIEVE command use.

**Syntax**

```
SET AUTOKEYRETRIEVE
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFF</td>
<td>the initial value for AUTOKEYRETRIEVE is OFF. Not specifying an argument for the command is equivalent to specifying OFF.</td>
</tr>
<tr>
<td>ROW</td>
<td>the identity column should be included along with the entire resulting row.</td>
</tr>
<tr>
<td>COLUMN</td>
<td>only the identity column associated with INSERT statements should be returned.</td>
</tr>
</tbody>
</table>

**Usage Notes**

<table>
<thead>
<tr>
<th>When Using the AUTOKEYRETRIEVE Command</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>prior to establishing sessions,</td>
<td>using the AUTOKEYRETRIEVE command prior to establishing sessions is allowed.</td>
</tr>
<tr>
<td></td>
<td>The current AUTOKEYRETRIEVE value will remain in effect for all subsequent requests until explicitly changed. It is persistent across LOGONs done within the same BTEQ invocation.</td>
</tr>
<tr>
<td></td>
<td>However, when the current database version does not support auto generated key retrieval, the setting will be ignored.</td>
</tr>
</tbody>
</table>
Example 1
This example shows the effects of having executed the AUTOKEYRETRIEVE OFF command.

Given the following table definition for an empty table:

```
CREATE TABLE MyTable
  , NO Fallback
  , NO BEFORE JOURNAL
  , NO AFTER JOURNAL
  
  (   MyCol1  INTEGER GENERATED ALWAYS AS IDENTITY
  , MyCol2  INTEGER
  )
  PRIMARY INDEX ( MyCol1 )
  ;
```

If the following SQL insert statement is submitted with AUTOKEYRETRIEVE set to OFF,

```
INSERT INTO MyTable VALUES(,1);
```

the result generated only shows an activity count and elapsed time status messages similar to the following:

```
*** Insert completed. One row added.
*** Total elapsed time was 1 second.
```

Example 2
This example shows the effects of having executed the AUTOKEYRETRIEVE COLUMN command.

Given the following table definition for an empty table:

```
CREATE TABLE MyTable
  , NO Fallback
  , NO BEFORE JOURNAL
  , NO AFTER JOURNAL
  
  (   MyCol1  INTEGER GENERATED ALWAYS AS IDENTITY
  , MyCol2  INTEGER
  )
  PRIMARY INDEX ( MyCol1 )
  ;
```

If the following SQL insert statement is submitted with AUTOKEYRETRIEVE set to COLUMN,

```
INSERT INTO MyTable VALUES(,1);
```

the result generated includes the value generated for the identity column.

```
*** Insert completed. One row added.
*** Total elapsed time was 1 second.

MyCol1
-------
2
```

Example 3
This example shows the effects of having executed the AUTOKEYRETRIEVE ROW command.
Given the following table definition for an empty table:

```sql
CREATE TABLE MyTable
    , NO FALBACK
    , NO BEFORE JOURNAL
    , NO AFTER JOURNAL
/
    
    MyCol1  INTEGER GENERATED ALWAYS AS IDENTITY
    , MyCol2  INTEGER
)
PRIMARY INDEX ( MyCol1 )
;
```

If the following SQL insert statement is submitted with AUTOKEYRETRIEVE set to ROW,

```sql
INSERT INTO MyTable VALUES(,1);
```

the result generated includes values for the entire row generated for the insert.

```sql
*** Insert completed. One row added.
*** Total elapsed time was 1 second.

MyCol1     MyCol2
----------- -----------
    3          1
```
## COMPILE

### Purpose
Defines (creates or replaces) a stored procedure object in the database using an SPL input file.

### Syntax

```
.COMPILE FILE 'filename' [WITH NOSPL] [filespec...
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename</td>
<td>name of a file containing source text for creating stored procedures on network-attached systems, or a DD name on channel-attached systems. For network-attached systems, if the name of the file includes a comma, semicolon, or space character, enclose the entire file name in single or double quotation marks. In the following example, <code>myproc.spl</code> is the file name: <code>.COMPILE FILE 'myproc.spl'</code> Also, any time text follows a file name, that file name must be enclosed in quotation marks. For example: `.COMPILE FILE 'myproc.spl' WITH NOSPL</td>
</tr>
<tr>
<td>NOSPL</td>
<td>indication that the stored procedure source text should not be stored in the Teradata Database server.</td>
</tr>
<tr>
<td>SPL</td>
<td>indication that the stored procedure source text needs to be stored in the Teradata Database server. If no argument is specified in the “With” part of the command, BTEQ assumes SPL.</td>
</tr>
<tr>
<td>WITH</td>
<td>beginning of the syntax element to specify the SPL storing option.</td>
</tr>
<tr>
<td>xxx</td>
<td>name of the z/OS JCL DD statement that defines the file from which SPL statements are read on channel-attached systems.</td>
</tr>
</tbody>
</table>
Note: Specify the dot (.) at the beginning, or the semicolon (;) at the end, or both, in the COMPILE command. If you do not include the dot character, BTEQ generates a warning message but executes the command.

Usage Notes
The COMPILE command can only be used for non-external stored procedures. For differences between external and non-external stored procedures, see “Creating and Using Stored Procedures” on page 72.

You must have CREATE PROCEDURE privilege on the database specified in the COMPILE command. For replacing or modifying an existing stored procedure using the REPLACE PROCEDURE statement in the stored procedure source text, you must have DROP privilege on the particular stored procedure or DROP PROCEDURE privilege on the database containing it.

The stored procedure definition in the input file begins with the keywords CREATE/REPLACE PROCEDURE, followed by the stored procedure body, and ends with a semicolon. The definition can also contain parameters, local variables, exception handlers and comments.

The following rules apply:
1. You must log on to Teradata Database before executing this command.
2. The source text for a stored procedure must be written in an input file. You cannot issue the CREATE/REPLACE PROCEDURE SQL statement at the BTEQ prompt.
3. One file can contain source text for only one stored procedure. It cannot contain any other BTEQ commands.
4. The COMPILE command must be the last statement in an open transaction.

If there are no compilation errors, then the entire stored procedure source text is stored in the specified database. It can be retrieved later using the SHOW PROCEDURE SQL statement.

For other details of the features and limitations of SPL, plus sample input files, refer to SQL Data Manipulation Language.

Note: For Unicode sessions, a BOM is optional at the beginning of a UTF8 or UTF16 file, and the encoding of a stored procedure file must match the encoding of the session character set.

SPL Attribute
You can also indicate to Teradata Database server whether to store or not to store the stored procedure source text.

Specify SPL if you want the stored procedure source text to be stored in Teradata Database. This is the default option.

NOSPL indicates that the stored procedure source text should not be stored in Teradata Database. For such stored procedures, the SHOW PROCEDURE SQL statement returns an error/failure.
**Session Mode**

The session mode (Teradata or ANSI) is stored with each stored procedure created in the database. This information is used during stored procedure execution.

A stored procedure created in a given mode can be executed only in the same mode. For example, a stored procedure created in ANSI mode can be executed only in ANSI mode.

This rule allows you to define a stored procedure with the transaction control statements that are specific to a given session mode, and to define defaults associated with a session mode.

**Unqualified Objects in the Source Text**

If any database objects referenced in the SQL statements inside the stored procedure source text are not qualified with a database name, it is assumed that the objects are in the current default database of the session in which you have created the stored procedure.

The unqualified objects are qualified before creating the stored procedure database object.

**Errors in COMPILE**

The COMPILE command results in a SUCCESS, OK, or FAILURE response from Teradata Database.

SPL compilation errors/warnings are reported as part of a SUCCESS/OK response; hence, they have no impact on an open transaction.

If there are any compilation errors, the stored procedure is not created or replaced.

If only compilation warnings occur, the stored procedure is created or replaced as requested.

If a failure is reported (in either ANSI or Teradata mode) as a response to the COMPILE command, the entire transaction is rolled back.

See “Handling Errors” on page 85 for details of the error response size and organization.

Refer to Messages for a complete list of the SPL compilation error and warning messages.

**Aborting Compilation**

If you abort a COMPILE command using the ABORT feature, the following rules apply:

- For CREATE PROCEDURE, the stored procedure is not defined in the database.
- For REPLACE PROCEDURE, the original stored procedure definition is retained.

**Compatibility Across Platforms**

A stored procedure executable is created on the Teradata Database server. The executable code is compatible only with the operating environment of the server. Hence, a stored procedure created on a Teradata Database platform (for example, Solaris) cannot be executed on another Teradata Database server platform, such as Microsoft Windows (assuming that the stored procedure is archived from Solaris) and restored on Windows.

**Example 1**

This example shows the correct usage of the COMPILE command:

```
.SET SESSION TRANS BTET
```
Example 2
This example requires a COMMIT/ROLLBACK/ABORT after the COMPIL command to close the transaction.

```
SET SESSION TRANS ANSI;
LOGON server/user, password;
SELECT date;
.COMPILE FILE spSample1;
COMMIT;
```

Example 3
This example is not valid because the COMPIL command is not the last request in an open transaction (it is followed by a SELECT SQL statement):

```
SET SESSION TRANS BTET;
LOGON server/user, password;
BT;
.COMPILE FILE spSample1;
SELECT date;
ET;
```
DECIMALDIGITS

Purpose
Overrides the precision specified by a CLI System Parameter Block (SPB) max_decimal_returned entry, or if that entry does not exist, indicates what the precision should be for decimal values associated with subsequently issued SQL requests for non-Field Mode responses.

The SHOW CONTROLS command can be used to determine the current DECIMALDIGITS setting.

If a max_decimal_returned entry has actually been established, the initial value for the setting is based on the SPB entry’s value.

Syntax

```
SET DECIMALDIGITS n DEFAULT
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>The range for n is 0...255 inclusive. DEFAULT can be used to indicate that the setting should be reset back to its initial value. Using a 0 value, or no value at all, is equivalent to using DEFAULT. Non-0 integer values will be subject to further validation by the database server.</td>
</tr>
<tr>
<td>DEFAULT</td>
<td>DEFAULT can be used to indicate that the setting should be reset back to its initial value.</td>
</tr>
</tbody>
</table>

Usage Notes
Using the DECIMALDIGITS command prior to establishing sessions is allowed. The current DECIMALDIGITS setting is ignored when the database version being used does not support the Max Decimal Precision feature.

The current DECIMALDIGITS value remains in effect for all subsequent requests until explicitly changed, and is persistent across LOGONs done within the same BTEQ invocation.
The maximum decimal digits to return applies to all of the record modes (Record, Indicator, and Multipart Indicator), but does not apply to Field Mode. In Field Mode, a CAST must be performed or the FORMAT clause must be used.

The difference in behavior between Field and Record/Indicator Modes occurs because Field Mode deals with formatted data, where either FORMAT or CAST can be easily used. In the Record/Indicator Mode, the data is in internal representation, such as, packed decimal for DECIMAL data returned to the mainframe, where compatibility is important.

The Indicator Mode behavior with .SET DECIMALDIGITS \( n \) is the same as doing an explicit CAST. See SQL Functions, Operators, Expressions, and Predicates for more information on truncation and rounding.

The Field Mode, behavior ignores .SET DECIMALDIGITS \( n \). To get the same effect in Field Mode as in Indicator Mode with .SET DECIMALDIGITS \( n \), perform either a CAST to DECIMAL\( (n,n) \), or perform a FORMAT clause with \( n \) digits specified.

**Example**

To CAST to DECIMAL\( (28,28) \):

```
.logon dbs5/dbc,dbc
select * from dbc.dbcinfo;
select cast(0.000000000100000000020000000003 as DECIMAL(28,28));
.logoff
```
**DEFAULTS**

**Purpose**
Resets BTEQ command options to the values that were set when BTEQ was first invoked.

**Syntax**
```
SET DEFAULTS
```

**Usage Notes**
You can use the DEFAULTS command in a Teradata SQL macro. The following BTEQ format commands are reset by the DEFAULTS command.

<table>
<thead>
<tr>
<th>BTEQ Command</th>
<th>Default Value</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORMCHAR</td>
<td>OFF (network-attached system)</td>
<td>For printer form feed control, FORMCHAR command is set to ON as default.</td>
</tr>
<tr>
<td></td>
<td>DEFAULT (channel-attached system)</td>
<td></td>
</tr>
<tr>
<td>LOGONPROMPT</td>
<td>ON</td>
<td></td>
</tr>
</tbody>
</table>

**System Defaults and Command Defaults**
The values listed are those in effect by default when you invoke BTEQ initially, or after executing a DEFAULTS command. When a command is used without an explicitly specified value, the values in effect might be different than the default.

To be certain of the defaults, use the SHOW CONTROLS command both before and after the DEFAULTS command to see both the old and new values.

Table 7: BTEQ Commands and DEFAULTS Relationship

<table>
<thead>
<tr>
<th>BTEQ Command</th>
<th>Affected by DEFAULTS Command?</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABORT</td>
<td>No</td>
</tr>
<tr>
<td>AUTOKEYRETRIEVE</td>
<td>Yes</td>
</tr>
<tr>
<td>DECIMALDIGITS</td>
<td>No</td>
</tr>
<tr>
<td>DEFAULTS</td>
<td>N/A</td>
</tr>
<tr>
<td>ECHOREQ</td>
<td>Yes</td>
</tr>
<tr>
<td>ERRORLEVEL</td>
<td>No</td>
</tr>
<tr>
<td>ERROROUT</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Table 7: BTEQ Commands and DEFAULTS Relationship (continued)

<table>
<thead>
<tr>
<th>BTEQ Command</th>
<th>Affected by DEFAULTS Command?</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXIT</td>
<td>No</td>
</tr>
<tr>
<td>EXPORT</td>
<td>Yes (format or limit)</td>
</tr>
<tr>
<td></td>
<td>No (file name)</td>
</tr>
<tr>
<td>EXPORTEJECT</td>
<td>Yes</td>
</tr>
<tr>
<td>FOLDLINE</td>
<td>Yes</td>
</tr>
<tr>
<td>FOOTING</td>
<td>Yes</td>
</tr>
<tr>
<td>FORMAT</td>
<td>Yes</td>
</tr>
<tr>
<td>FORMCHAR</td>
<td>Yes</td>
</tr>
<tr>
<td>GOTO</td>
<td>No</td>
</tr>
<tr>
<td>HALT EXECUTION</td>
<td>No</td>
</tr>
<tr>
<td>HANG</td>
<td>No</td>
</tr>
<tr>
<td>HEADING</td>
<td>Yes</td>
</tr>
<tr>
<td>HELP BTEQ</td>
<td>No</td>
</tr>
<tr>
<td>IF...THEN...</td>
<td>No</td>
</tr>
<tr>
<td>LABEL</td>
<td>No</td>
</tr>
<tr>
<td>LARGEDATAMODE</td>
<td>Yes</td>
</tr>
<tr>
<td>LOGOFF</td>
<td>No</td>
</tr>
<tr>
<td>LOGON</td>
<td>No</td>
</tr>
<tr>
<td>LOGONPROMPT</td>
<td>Yes</td>
</tr>
<tr>
<td>MAXERROR</td>
<td>No</td>
</tr>
<tr>
<td>NULL</td>
<td>Yes</td>
</tr>
<tr>
<td>OMIT</td>
<td>No</td>
</tr>
<tr>
<td>PAGEBREAK</td>
<td>Yes</td>
</tr>
<tr>
<td>PAGELength</td>
<td>Yes</td>
</tr>
<tr>
<td>QUIET</td>
<td>Yes</td>
</tr>
<tr>
<td>QUIT</td>
<td>No</td>
</tr>
<tr>
<td>RECORDMODE</td>
<td>Yes</td>
</tr>
<tr>
<td>REMARK</td>
<td>No</td>
</tr>
<tr>
<td>REPEAT</td>
<td>No</td>
</tr>
<tr>
<td>REPEATSTOP</td>
<td>Yes</td>
</tr>
</tbody>
</table>
## Chapter 5: BTEQ Commands

### DEFAULTS

**Resetting Defaults by Logging Off**

The LOGON and LOGOFF commands do not reset the format command default values. If you log off from the initial BTEQ session and then specify another LOGON command, the resulting sessions inherit the format command values from the prior session. You must use the DEFAULTS command if you want to reset the format commands to their initial default values.

The only times that you can assume the format commands are at the values listed above are when you execute the first LOGON command after invoking BTEQ, and when the DEFAULTS command is used.

---

### Table 7: BTEQ Commands and DEFAULTS Relationship (continued)

<table>
<thead>
<tr>
<th>BTEQ Command</th>
<th>Affected by DEFAULTS Command?</th>
</tr>
</thead>
<tbody>
<tr>
<td>REPORTALIGN</td>
<td>Yes</td>
</tr>
<tr>
<td>RETCANCEL</td>
<td>Yes</td>
</tr>
<tr>
<td>RETLIMIT</td>
<td>Yes</td>
</tr>
<tr>
<td>RETRY</td>
<td>Yes</td>
</tr>
<tr>
<td>RTITLE</td>
<td>Yes</td>
</tr>
<tr>
<td>RUN</td>
<td>No (file name) Yes (skip)</td>
</tr>
<tr>
<td>SEPARATOR</td>
<td>Yes</td>
</tr>
<tr>
<td>SESSION CHARSET</td>
<td>No</td>
</tr>
<tr>
<td>SESSION SQLFLAG</td>
<td>No</td>
</tr>
<tr>
<td>SESSION TRANSACTION</td>
<td>No</td>
</tr>
<tr>
<td>SHOW CONTROLS</td>
<td>N/A</td>
</tr>
<tr>
<td>SHOW VERSIONS</td>
<td>N/A</td>
</tr>
<tr>
<td>SIDETITLES</td>
<td>Yes</td>
</tr>
<tr>
<td>SKIPDOUBLE</td>
<td>Yes</td>
</tr>
<tr>
<td>SKIPLINE</td>
<td>Yes</td>
</tr>
<tr>
<td>SUPPRESS</td>
<td>Yes</td>
</tr>
<tr>
<td>TDP</td>
<td>No</td>
</tr>
<tr>
<td>TITLEDASHES</td>
<td>Yes</td>
</tr>
<tr>
<td>TSO</td>
<td>No</td>
</tr>
<tr>
<td>UNDERLINE</td>
<td>Yes</td>
</tr>
<tr>
<td>WIDTH</td>
<td>Yes</td>
</tr>
<tr>
<td>=</td>
<td>No</td>
</tr>
</tbody>
</table>

---

*Basic Teradata Query Reference*
Chapter 5: BTEQ Commands

The following list shows the initial default condition of BTEQ at the first LOGON. When the DEFAULTS command is used, it resets these formats to the initial default value.

show controls;
Default Maximum Byte Count = 1048500
Default Multiple Maximum Byte Count = 2048
Current Response Byte Count = 32705
Maximum number of sessions = 200
Maximum request size = 32705
Maximum IMPORT/USING data size = 32705
Maximum number of returnable columns = 2048
Maximum number of WITH clauses = 10
Maximum number of times in a REPEAT = 2147483647
Maximum number of script files = 30
Maximum number of lines in a TITLE = 10
Maximum number of lines per page = 2147483647
Maximum string length = 254
Maximum number of WIDTH in a report = 65531
Minimum number of WIDTH in a report = 20

Maximum Notify MSG text Byte Count = 254
Maximum number of bytes saved for SUPPRESS, PAGEBREAK, SKIPLINE, UNDERLINE, or SKIPDOUBLE commands, or for &n substitutions = 256

Client Platform Byte Order = LITTLE ENDIAN

EXPORT  RESET
IMPORT FIELD
LOGMECH = default
LOGON
REPEAT = 1
RUN
[SET] AUTOKEYRETRIEVE = OFF
[SET] DECIMALDIGITS = 0 (SPB DEFAULT)
[SET] ECHOREQ = ON
[SET] ENCRYPTION = OFF
[SET] ERRORLEVEL = ON
[SET] FOLDLINE = OFF ALL
[SET] FOOTING = NULL
[SET] FORMAT = OFF
[SET] FORMCHAR = OFF
[SET] FULLYEAR = OFF
[SET] HEADING = NULL
[SET] INDICDATA = OFF
[SET] LARGEDATAMODE = OFF
[SET] LOGONPROMPT = ON

[SET] NOTIFY = OFF
[SET] NULL = ?
[SET] OMIT = OFF ALL
[SET] PACK = 0
[SET] PAGEBREAK = OFF ALL
[SET] PAGELENGTH = 55
[SET] PREPAREMODE = OFF
[SET] QUIET = OFF
[SET] RECORDMODE = OFF
[SET] REPEATSTOP = OFF
[SET] REPORTALIGN = COMPATIBLE
[SET] RETCANCEL = OFF
[SET] RETLIMIT = Rows: No Limit Columns: 100
[SET] RETRY = ON
[SET] RTITLE = NULL
[SET] SEPARATOR = two blanks

[SET] SESSION CHARSET = ASCII
import/export encoding = ASCII
stdin/stdout encoding = ASCII
[SET] SESSION RESPBUFLEN = 8192
[SET] SESSION SQLFLAG = NONE
[SET] SESSION TRANSACTION = Unknown
[SET] SESSION TWORESPBUFS = ON
[SET] SESSIONS = 1
[SET] SIDETITLES = OFF for the normal report.
And, it is ON for results of WITH clause number: 1 2 3 4 5 6 7 8 9.
[SET] SKIPDOUBLE = OFF ALL
[SET] SKIPLINE = OFF ALL
[SET] SUPPRESS = OFF ALL
[SET] TDP = dbc
[SET] TIMEMSG = DEFAULT
[SET] TITLEDASHES = ON for the normal report.
And, it is ON for results of WITH clause number: 1 2 3 4 5 6 7 8 9.
[SET] UNDERLINE = OFF ALL
[SET] WIDTH = 75

**Example 1**
To reset the BTEQ format commands to their default values, type:

```
.SET DEFAULTS
```

**Example 2**
The Example 1 command in a Teradata SQL macro appears as:

```
ECHO '.SET DEFAULTS';
```
ECHOREQ

**Purpose**
Specifies whether a copy of the Teradata SQL requests and BTEQ commands should be placed in the standard output stream.

**Syntax**
```
SET ECHOREQ [ON | OFF]
```

**Usage Notes**
Use the ECHOREQ command before a series of BTEQ commands or requests to enable or inhibit the return (echo) of a copy of each command or request. (BTEQ does not echo the password portion of the LOGON command.)

<table>
<thead>
<tr>
<th>If</th>
<th>Then</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECHOREQ command has not been specified,</td>
<td>The initial default value is OFF</td>
</tr>
<tr>
<td>ECHOREQ command is used and ON or OFF is not specified,</td>
<td>BTEQ assumes ON</td>
</tr>
</tbody>
</table>

**Note**: The ECHOREQ command is not related to the Teradata SQL ECHO statement. You cannot use the ECHOREQ command in a Teradata SQL macro.

**Example**
To inhibit echoing input, type:
```
.SET ECHOREQ OFF
```
**Purpose**

The SET ENCRYPTION command toggles request-level Full-Stream Encryption ON or OFF. The initial value for the setting is based on the value of CLI’s clispb.dat file data_encryption entry. When the command is used with neither ON or OFF specified, the ON value is assumed by default.

Choosing which requests are encrypted and which are sent in clear text format is important because encryption can be an expensive task for processor resources and elapsed time.

If the ERRORLEVEL command has not been used, its value is ON by default.

**Syntax**

```
SET ENCRYPTION [ON | OFF]
```

where

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td>Encryption is turned on</td>
</tr>
<tr>
<td>OFF</td>
<td>Encryption is turned off</td>
</tr>
</tbody>
</table>

**Usage Notes**

If a client application wants a request encrypted, the response to that request is also encrypted. This function is important to BTEQ users transferring client-side files to servers to create User defined Functions, Types, or Stored Procedures. Turning ON encryption for these requests helps prevent a “man in the middle” scenario in which a network interloper can concentrate his or her efforts on specific transmissions based on the contents of the corresponding request or response.

Since BTEQ does not parse SQL, users, particularly those performing DBA tasks, should use encryption whenever SQL requests are submitted that could otherwise inadvertently disclose a password.

BTEQ automatically enables encryption for one specific interactive mode logon scenario. If BTEQ has detected that the DBS returned notification that the user’s password has expired, a prompt is given to enter a new password so that BTEQ can construct and submit a MODIFY USER statement for the user. The password update request is encrypted. BTEQ then goes back to using the pre-existing encryption setting.
**ERRORLEVEL**

**Purpose**
Assigns severity levels to errors. Use the severity levels to determine a course of action based on the severity of the errors that BTEQ encounters.

If the ERRORLEVEL command has not been used; its value is ON by default.

**Syntax**

```
ERRORLEVEL

set

errno

( errno )

ON

OFF

UNKNOWN

n

SEVERITY
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>errno</code></td>
<td>Specification of the error number</td>
</tr>
<tr>
<td><code>n</code></td>
<td>Specification of the assigned severity level, where <code>n</code> = 2, 4, 8, or 12</td>
</tr>
<tr>
<td><code>ON</code> or <code>OFF</code></td>
<td>Enable or disable specification for the error level function</td>
</tr>
<tr>
<td><code>UNKNOWN</code></td>
<td>Designation of other BTEQ error codes not defined by default mapping or by you</td>
</tr>
</tbody>
</table>

**Usage Notes**
Normally, you would use a BTEQ script to query the status of the last Teradata SQL statement and obtain information such as the number of rows involved, and any error code that might have been returned by Teradata Database. You can use the ERRORLEVEL command to extend that functionality by assigning a severity level to each Teradata SQL error code.

You can also use the ERRORLEVEL command to assign severity levels to the return codes that BTEQ returns to the operating system when a job completes. In this way you can correlate ambiguous system-assigned error numbers to meaningful levels of error severity for alerting users. (For more information on using return codes in this manner, see “Handling Errors” on page 85.

The ERRORLEVEL command includes one or more error items. Each error item consists of: either the word UNKNOWN, a single error number, or a parenthetical list of error numbers,
followed by the word SEVERITY, followed by an integer value. The integer value is the new error level value for the listed error numbers.

(The error code of UNKNOWN accommodates any other BTEQ error codes that are not defined.)

In addition to the ON and OFF ERRORLEVEL command options, you can use the following system variables to enable or disable the BTEQ error level function:

- The z/OS parameter string
- UNIX environment variables

The following example redefines the level of several errors and tests the result of a query using the IF..THEN... command.

```
.SET ERRORLEVEL 2168 SEVERITY 4, (2173, 3342, 5262) SEVERITY 8
.SET ERRORLEVEL UNKNOWN SEVERITY 16
...
SELECT * FROM SOMEWHERE;
.IF ERRORLEVEL >= 14 THEN .QUIT 17;
...
**ERROROUT**

**Purpose**
Routes the standard error stream and the standard output stream to two files or devices for channel-attached systems, or to one file or device for network-attached client systems.

**Syntax**

```
SET ERROROUT STDERR
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Directs Error Messages to</th>
</tr>
</thead>
<tbody>
<tr>
<td>STDERR</td>
<td>Default and normal output file for network-attached client systems, called STDERR</td>
</tr>
<tr>
<td>STDOUT</td>
<td>Default and normal output file for channel-attached systems, called STDOUT</td>
</tr>
</tbody>
</table>

**Usage Notes**
The ERROROUT command affects error and output streams differently for channel-attached client and network-attached systems.

When you specify STDERR on a channel-attached system, you split errors and output to two separate files or devices.

When you specify STDOUT on a network-attached system, you merge both STDERR and STDOUT to one file or device.

**Note:** PC users can direct errors and output to one file or device by specifying STDOUT.

**Note:** If STDERR is specified in a z/OS batch environment on a channel-attached system, BTEQ writes your error messages to the ddname SYSTERM. If STDERR is specified in a z/OS/TSO environment, BTEQ displays your error messages on the terminal.

You can use the ERROROUT command in a Teradata SQL macro.

**Example 1**
In this example, the standard error stream and standard output stream on a channel-attached system are directed to one file:

```
BTEQ -- Enter your Teradata SQL request or BTEQ command:
.set errorout stderr
.set errorout stderr
*** Error messages now directed to STDERR.
```
Example 2
The Example 1 command in a Teradata SQL macro appears as:

    ECHO '.SET ERROROUT STDERR';
EXIT

**Purpose**
Logs off all Teradata Database sessions and exits BTEQ.

**Syntax**

```
EXIT
QUIT n
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTIVITYCO</td>
<td>The current value of the ACTIVITYCOUNT status value</td>
</tr>
<tr>
<td>ERRORCODE</td>
<td>The error code generated by the last Teradata SQL request as the job step return code</td>
</tr>
<tr>
<td>ERRORLEVEL</td>
<td>The highest error severity level that you assigned</td>
</tr>
<tr>
<td><code>n</code></td>
<td>A decimal number as the job step return code. To avoid existing codes, do not use numbers from 1 to 10.</td>
</tr>
</tbody>
</table>

**Usage Notes**
The BTEQ EXIT and QUIT commands are identical. For your own convenience in writing scripts, you may prefer to use the EXIT command after an explicit LOGOFF command and use QUIT if the LOGOFF command is not explicitly provided.

You cannot use the EXIT command in a Teradata SQL macro.

The ERRORCODE value contains only the status of the last statement, even in a set of statements specified with the REPEAT command.

**Example**
To exit BTEQ, type:

```
EXIT
```
**Purpose**

Specifies the name and format of an export file that BTEQ will use to store database information returned by a subsequent SQL SELECT statement.

**Note:** The EXPORT command should not be run on the Teradata Database console window of the AWS. This is because the output file specification specifies the location of the file on the current TPA (trusted parallel application) control node and will appear on this current node and not on the AWS where it is being run.

**Syntax**

For Channel-Attached Systems:

```
.EXPORT DATA INDICDATA REPORT DDNAME DDNAME FILE ddname

, LIMIT = n1 , OPEN CLOSED

, NOEJECT EJECT BOM NOBOM

DIF LABELS

RESET
```

- **FILE** 
  - Specifies the name of the file that will be used for the export.
  - Format: `filename`

- **DDNAME** 
  - Specifies the data set name.
  - Format: `ddname`

- **LIMIT** 
  - Specifies the number of rows to be included in the export.
  - Format: `n1` (optional), `n2` (optional)
For Network-Attached Systems:

```
EXPORT

\[ . \text{EXPORT} \quad \text{DATA} \quad \text{AXSMOD} \quad \text{REPORT} \quad \text{BOM | NOBOM} \quad \text{DIF} \quad \text{BOM | NOBOM} \quad \text{FILE} \quad \text{filename} \quad \text{LIMIT = n1} \quad \text{LIMIT = n2} \quad \text{OPEN} \quad \text{CLOSED} \quad \text{AXSMOD} \quad \text{modname} \quad \text{‘init-string’} \quad \text{RESET} \]
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>AXSMOD</td>
<td>Indicates that data is to be exported using an access module. The file size can be greater than 2 GB. This option is only valid for DATA and INDICDATA types on network-attached systems.</td>
</tr>
<tr>
<td>BOM</td>
<td>NOBOM</td>
</tr>
<tr>
<td>DATA</td>
<td>Returns the results of a query to a file for processing by other programs. BTEQ returns data in Record Mode and stores it in FastLoad format output file. Use this format to interchange data between BTEQ, FastLoad, and FastExport. MultiLoad also accepts the FastLoad format. For more information about Record Mode, refer to “RECORDMODE” on page 246.</td>
</tr>
</tbody>
</table>
### Syntax Element: DATALABELS

Includes the column titles of the selection results as the first data row in the DIF file. The DATALABELS option is not available on network-attached systems.

**Note:** Network-attached systems ignore the BTEQ TITLE DASHES command if the export file uses DIF format.

### Syntax Element: ddname

Specifies the name of the z/OS JCL DD statement that defines the file specified by the EXPORT command for receiving the results from the subsequent SQL SELECT statement.

**Note:** SYSPRINT is not a valid DDNAME for the BTEQ EXPORT command. You cannot export to the SYSPRINT file.

**Note:** Though you can use the keyword FILE instead of the keyword DDNAME for channel-attached systems, be very careful when you do because the implementation is for the concept DDNAME, not the concept FILE.

**Note:** If you do not define an export file name, the default field for the exported data is FILE ddname.

**Note:** When the ddname specified for mainframe BTEQ is not associated with a physical data set that is defined, the corresponding error message will be printed and BTEQ terminates if running batch mode or continues execution if running in interactive mode.

**Note:** On IBM z/OS, BTEQ does not support exporting a REPORT/DIF to a data set with ASA carriage control character supported record format; for example, RECFM=VBA, FA, FBA, and so on for Unicode as active session charset.

### Syntax Element: DIF

Converts the results of a query to Data Interchange Format (DIF), a text format used by many spreadsheet applications for importing and exporting data.

**Note:** This parameter is not supported under the Kanji character sets KANJISJIS_0S or KANJIEUC_0U, or the Chinese character sets TCHBIG5_1R0 or SDTCHBIG5_3R0.

### Syntax Element: EJECT

Generates an additional page advance carriage control character at the top of the EXPORT file using the REPORT keyword. This works only for DISP=(OLD,NEW) on z/OS.
Chapter 5: BTEQ Commands

**EXPORT**

filename

Names the network-attached system file that receives the results of the SQL SELECT statement.

If the name of the file includes a comma, semicolon, or space character, enclose the entire file name in single or double quotation marks.

Also, note that file names are case-sensitive on systems running under UNIX, and they are not case-sensitive on systems running under Windows.

Whenever other command options are specified following the file name in the Windows environment, BTEQ requires the file name to be enclosed by single or double quotation marks.

**Note:** Though you can use the keyword ddname instead of the keyword FILE for network-attached systems, be very careful when you do because the implementation is for the concept FILE, not the concept DDNAME.

**Note:** When the filename specified for工作站 BTEQ exists and is read-only, or the PATH specified to EXPORT the file does not exist, the corresponding error prints. When running in batch mode, BTEQ terminates after printing the error message. When running in interactive mode, BTEQ continues execution after printing this error message.

INDICDATA

Returns the results of a query to a file using indicator variables to identify null values.

BTEQ returns data in Indicator Mode and stores it in the appropriate format.

**Note:** INDICDATA is the only valid data format specification if the LARGEDATAMODE setting is ON.

For more information about Indicator Mode, refer to “INDICDATA” on page 197.

‘init-string’

Names the initiation string used to start the import process.

modname

Names the file that is to receive the imported data.

n1

Is the maximum number of rows that Teradata Database can return to the specified export file.

Because the BTEQ RETLIMIT command also sets this specification, the most recent LIMIT=n1 option or RETLIMIT command overrides the previous specification.

The maximum limit that you can specify is:

- 2,147,483,407 for channel-attached systems
- 2,147,483,647 for network-attached systems

The default value is n1=0, which represents no limit.

(For details, refer to “RETLIMIT” and “RETCANCEL” in this chapter.)
Chapter 5: BTEQ Commands

**EXPORT**

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>n2</em></td>
<td>Is the maximum number of columns that Teradata Database can return to the specified export file. <strong>Note:</strong> You must enter a value for <em>n1</em> before you can enter a value for <em>n2</em>. The value of <em>n2</em> can be an asterisk or a number between 1 and 2048. An * signifies the maximum columns that BTEQ currently supports. This value is subject to change to support future releases of Teradata Database. If you want to fix the limit of columns that are returned, enter a number instead of an asterisk. For the current Teradata Database limit, refer to <em>SQL Fundamentals</em>, Appendix C. Before you increase the column limit, you might need to reestablish other settings so that the effect of the setting is populated over the additional columns. For example, if you want to omit all but the third column, use the OMIT ON ALL command before you change the column limit. After you change the column limit, use the OMIT ON 3 command. Otherwise, the ON attribute will not be populated for the additional columns.</td>
</tr>
<tr>
<td><strong>NOEJECT</strong></td>
<td>Suppresses the generation on an additional page advance character at the top of the EXPORT file using the REPORT keyword. This works only for disposition type DISP=(OLD,NEW) on z/OS.</td>
</tr>
<tr>
<td><strong>OPEN</strong> or <strong>CLOSE</strong></td>
<td>Determines the disposition of the export file in the event of a Teradata Database restart. If you specify the OPEN option and a restart occurs, BTEQ starts the query again and adds rows to the end of the currently open output file. (BTEQ resubmits the query because ON is the default setting of the RETRY command.) This duplicates the rows that were exported before the restart. If you specify the CLOSE option with the RETRY command option set to OFF and a restart occurs, BTEQ does not resubmit the query. In this case, the query results are lost because the output file is empty or contains inconsistent data, only some of the returned rows.</td>
</tr>
<tr>
<td><strong>REPORT</strong></td>
<td>Outputs a query result to the previously defined file. This is the default. For z/OS, the lines in the export file are truncated to 253 characters by default. The width limit can be increased to 65531 characters using the WIDTH command. The REPORTWIDE keyword is also accepted. This keyword specifies that the REPORT format for the data should be used, and changes the WIDTH setting to 32765. Because this keyword is deprecated, a SET WIDTH command should be used instead. See the WIDTH command “WIDTH” on page 322.</td>
</tr>
<tr>
<td><strong>RESET</strong></td>
<td>Disables the previous EXPORT command and sends selection results to the standard output stream. If WIDTH was changed during the EXPORT, it is reset to its original value, the value that was in effect before that EXPORT.</td>
</tr>
</tbody>
</table>

**Usage Notes**

If the file being exported using **EXPORT** command of BTEQ is destined to serve as the source for **IMPORT** command across a different platform type, ensure that the endianness type of
both platforms is the same. This can be verified from the “Client Platform Byte Order” tab in the output of **SHOW CONTROLS** command.

BTEQ opens the specified export file when Teradata Database begins returning data—not when the **EXPORT** command is used. BTEQ also automatically opens an export file if Teradata Database returns any of the following parcels in response to an SQL statement:

- OK
- Success
- Failure
- Error

**Note:** BTEQ does not open an export file if Teradata Database returns an error parcel.

**Note:** For information on I/O errors and abends, refer to “I/O Errors and Abends” on page 92.

If you specify an existing export file, BTEQ appends new records after the existing records. If errors exist, however, then the standard output stream with errors is directed to the file specified by the BTEQ **ERROROUT** command.

You can use any device name that is valid on your system as an export file name. Each row returned by the Teradata SQL SELECT statement generates one record in the export file.

You cannot use the **EXPORT** command in a Teradata SQL macro.

### Suppressing Title Text and Titledashes in EXPORT REPORT Output

The following example demonstrates how to suppress the generation of column headers and titledashes at the top of an exported report.

```
.LOGON mydbs/myid, mypwd
CREATE TABLE Employer_Location (EmpNo INTEGER, Location CHAR(5));
INSERT INTO Employer_Location VALUES(1, 'LA');
INSERT INTO Employer_Location VALUES(2, 'NYC');
INSERT INTO Employer_Location VALUES(3, 'ATL');
.SET TITLEDASHES OFF
.EXPORT REPORT FILE=myfile1.exp
SELECT EmpNo (TITLE ''), Location (TITLE '') FROM Employer_Location;
.EXPORT RESET
.SET TITLEDASHES ON
.EXPORT REPORT FILE=myfile2.exp
SELECT EmpNo, Location FROM Employer_Location;
.EXPORT RESET
.LOGOFF
.EXIT
```

Output for **myfile1.exp**:

```
  1  LA
  2  NYC
  3  ATL
```

Output for **myfile2.exp**:

```
  EmpNo  Location
  --------  ------
      1      LA
```
Note: For file *myfile1.exp*, with TITLEDASHES turned OFF and the phrase TITLE "" (title text NULL) used for all columns in the SELECT statement, BTEQ recognizes that there is no header information to print and immediately begins printing data.

Note: For file *myfile2.exp*, with TITLEDASHES turned ON and no phrase for TITLE used with the columns in the SELECT statement, BTEQ prints the column header and titledashes lines and then begins printing data.

Note: After suppressing the title text and titledashes in the exported report, you can use the IMPORT command with data format REPORT on this file without making any modifications.

**SELECT Statement Response in Record Mode**

If the SQL SELECT statement generates its response in Record Mode (keyword DATA) or Indicator Mode (keyword INDICDATA), BTEQ passes each row on to the export file as a record. On network-attached systems, each record has the following format:

- The record begins with a length field, which is two bytes long and contains an unsigned binary integer whose value is the number of bytes in the record (not including the bytes in the length field and not including the byte(s) in the end-of-record field).
- The record follows with binary-formatted fields that contain the column values of the selected row in record or indicator format.
- The record ends with an end-of-record (line) indicator, appropriate for the workstation and operating system. For example, for UNIX, the indicator is a new line. For Windows, the indicator is a carriage return/line feed pair. In all cases, the end-of-record character, or characters, are in the bit-format appropriate for the workstation. For example, in ASCII for an ASCII workstation.

On channel-attached systems, each record has the following format:

The record consists of binary-formatted fields that contain the column values of the selected row in record or indicator format.

**Character Constants Treated as Variable-Length Strings (Record Mode)**

When a character constant is included in the SELECT statement's item list, the constant is treated as a variable-length string. In Record Mode, this causes the constant to be prefixed with a two-byte field that defines the length of the character constant. This behavior affects how a record containing such constants is defined when imported later by BTEQ or another program.

To export a fixed-length character string, use a FORMAT phrase in the SQL SELECT statement. For example, the FORMAT phrase in the following SELECT statement causes the character constant 'First' to return as a fixed-length 30-character string:

```
SELECT 'First' (FORMAT 'X(30)');
```
**SELECT Statement Response in Field Mode**

If the SQL SELECT statement generates its response in Field Mode (keyword REPORT), BTEQ passes each line of the report on to the export file as a record with the following format:

- The record begins with either a heading line or text-formatted fields that contain the column values of the selected row. In either case, the information is in the bit-format appropriate for the workstation and operating system (for example, in ASCII for an ASCII workstation).

- The record ends with an end-of-record (line) indicator, appropriate for the workstation and operating system. For example, for UNIX, the indicator is a new line. For Windows, the indicator is a carriage return/line feed pair. In all cases, the end-of-record character, or characters, are in the bit-format appropriate for the workstation. For example, in ASCII for an ASCII workstation.

**Note:** If you are going to edit the exported records on a PC and import them back to the database, be sure to use a PC editor that recognizes both the carriage return and line feed characters as an end-of-record marker.

When performing any export operation, you do not need to set the RECORDMODE command option; it is set by the executing EXPORT statement. (Use the SHOW CONTROLS command if you want to see the current response mode.)

**Closing the Export File**

BTEQ closes the export file when one of the following is used:

- An EXPORT command specifying either another export file or the RESET option
- A LOGOFF or EXIT/QUIT command

BTEQ does not close the file until the last file record has been written. If you read the file before has been closed, the last few records might be missing. They will appear when the file is closed in response to your EXPORT, LOGOFF, or EXIT/QUIT command.

In z/OS, if you specify a disposition of MOD in the JCL, the CLOSE option does not work.

**Channel-Attached Systems**

The following information applies only to channel-attached systems. If you have set LRECL and blocksize too small for your file, your data is shortened to those settings and Teradata Database returns the following error message:

```
***Increase test file record length and blocksize to match data.
```

BTEQ supports all QSAM-compatible formats. When you use the REPORT option, the DCB for the DD statement should specify a RECFM of VA, VBA, FBA, or FA to preserve page ejects.

When you use the DATA or INDICDATA options, the DCB for the DD statement can specify any RECFM; however, RECFM=FB or RECFM=VB are commonly used.

Regardless of the output option used, if RECFM is fixed (F, FB, FA, or FBA), then LRECL must be exact (including indicator bits, if any). Therefore, a variable RECFM (V, VA, VB, or VBA) with a larger record/blocksize is recommended, especially when the REPORT option is used or when a multi-byte session character set is used.
For example:

    (\// DCB=(RECFM=VB,BLKSIZE=32760,DSORG=PS)

Also, when using the EXPORT DATA, EXPORT INDICDATA, or EXPORT REPORT forms of the EXPORT command, the DCB for the receiving file should enter LRECL and BLKSIZE values that are big enough to accommodate the largest record that you expect to have.

You can append records to an existing file by entering a disposition of MOD for the file.

For z/OS BTEQ, the WIDTH of the SYSPRINT file temporarily changes with EXPORT REPORT/DIF in progress to match the LRECL value of the EXPORT file. This behavior is observed only so long as the LRECL of the EXPORT file is less than the current width of the SYSPRINT file. Also, this WIDTH gets reset to its original value once an EXPORT RESET command is executed.

**Example 1**

In Record Mode, to send the results of a SELECT statement to the data set allocated to the file SAVEDATA, type:

    .EXPORT DATA FILE=SAVEDATA

**Example 2**

To include indicator variables in the result of an SQL SELECT statement sent to the dataset ALLDATA, type:

    .EXPORT INDICDATA DDNAME=ALLDATA

**Example 3**

To include indicator variables in the result of an SQL SELECT statement sent to the file defined by ALLDATA, type:

    .EXPORT INDICDATA DDNAME=ALLDATA ,CLOSE

**Example - AXSMOD**

This example exports data to a file named EXPORT.DAT; the data exported to this file can be more than 2 GB:

    .EXPORT DATA FILE = 'EXPORT.DAT' AXSMOD
**EXPORTEJECT**

**Purpose**
Enables suppression of the additional Page Advance ASA Carriage Control Character at the top of the EXPORT file using the REPORT keyword for z/OS BTEQ.

This command is not valid on network-attached systems. The command only works for disposition type DISP=(OLD,NEW) on z/OS.

**Syntax**

```
.SET EXPORTEJECT [ON | OFF]
```

**Usage Notes**
Use the EXPORTEJECT command in a BTEQ script to suppress the additional page advance character for all the following EXPORT’s using the REPORT keyword. The option EJECT/NOEJECT specified in the EXPORT command has precedence over the value of EXPORTEJECT command for the duration of the EXPORT.

<table>
<thead>
<tr>
<th>If</th>
<th>Then</th>
</tr>
</thead>
<tbody>
<tr>
<td>the EXPORTEJECT command has not been specified,</td>
<td>the initial default value is ON.</td>
</tr>
<tr>
<td>the EXPORTEJECT command is used, and ON or OFF is not specified,</td>
<td>BTEQ assumes ON</td>
</tr>
</tbody>
</table>

**Example 1**
The following example will result in the suppression of the additional page advance in the EXPORT REPORT file when EXPORTEJECT command is reset at the script level and the keywords EJECT/NOEJECT are not used with the EXPORT command:

```
.SET EXPORTEJECT OFF;
.EXPORT REPORT DDNAME=MYDD;
SELECT date;
.EXPORT RESET;
```
Example 2

The following example will result in the generation of the additional page advance in the EXPORT REPORT file due to precedence of EXPORT command’s "NOEJECT" keyword over the value of the EXPORTEJECT command:

```
.SET EXPORTEJECT ON;
.EXPORT REPORT DDNAME=MYDD;
SELECT date;
.EXPORT RESET;
```
Purpose
Splits each line of a report into two or more lines after the specified columns. This compresses the report into a smaller width.

Syntax

```
SET OFF ON n
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n )</td>
<td>one or more column numbers, ordered from left to right and separated by commas. The value of ( n ) can range from 1 to 2048.</td>
</tr>
</tbody>
</table>

Note: Though using a comma as a separator character between column numbers is the preferred construction, you can use a single space character instead of a comma as a separator character in the BTEQ FOLDLINE command.

Usage Notes
You can use the FOLDLINE command in a Teradata SQL macro.

The WIDTH command specifies the maximum number of characters for each line in your report. BTEQ limits the width of a line of printed output to the WIDTH command maximum. If your returned report rows will not fit within this limit, they will be truncated. To avoid this, use the FOLDLINE command to establish folding points so that BTEQ can use more than one report line to print each row. After printing the value for a column that has a FOLDLINE setting of ON, BTEQ starts a new line. If the row has remaining columns to be printed, their values will be printed on the next line.

BTEQ offsets each split (i.e., folded) line belonging to a specific row, two character positions from the beginning character position for the report. After all the columns for a row have been handled, BTEQ starts forming the output lines for the next row.

For example, consider the three following sample result lines. The lines have four columns and were printed without FOLDLINE:

```
A1A1  B1B1  C1C1  D1D1
A2A2  B2B2  C2C2  D2D2
A3A3  B3B3  C3C3  D3D3
```
If FOLDLINE was set to ON for the first column, the width of the printed lines would be smaller, and the values could be printed in the following format:

A1A1  B1B1  C1C1  D1D1
A2A2  B2B2  C2C2  D2D2
A3A3  B3B3  C3C3  D3D3

Although you can fold a row in several places, too much folding may produce a report that appears to be non-columnar and therefore could be confusing to read. The other factor to consider with FOLDLINE use is the way BTEQ tries to limit the number of output lines used to print each row. In addition to instructing BTEQ to fold, rather than truncate, results data, FOLDLINE ON settings cause BTEQ to discard folded lines that are comprised solely of space characters. If a line formed to the left of the split consists solely of spaces, BTEQ does not print it.

As long as the first column for the row has its foldline setting turned on, BTEQ will also discard the line formed to the right of the row’s last split when that line is comprised of all spaces. Therefore, when you have rows with folded blank values, use FOLDLINE settings carefully to prevent potential misinterpretation of resulting reports. You may need to use other commands in combination with FOLDLINE, such as UNDERLINE or SIDETITLES, to produce your report. The UNDERLINE command can provide visual row boundaries. Typically, however, the SIDETITLES command is used in conjunction with the FOLDLINE command to clearly label values so as to avoid ambiguous reports.

Using FOLDLINE with SUPPRESS Command
Use the FOLDLINE command with the SUPPRESS command to eliminate the blank space caused by suppressing sort columns that do not change very often.

Using FOLDLINE with PAGEBREAK Command
You can also use the FOLDLINE command with the PAGEBREAK command. In this case, set the page break column number to break on the first folded line that you set with the FOLDLINE command. If the page break column of the PAGEBREAK command does not occur on the first folded line, BTEQ splits the rest of the row onto the next page. For example:

```
.SET PAGEBREAK ON 1;
.SET FOLDLINE ON 1, 2, 3;
```

Default Values
If a FOLDLINE command is not specified, its values are OFF and ALL by default. If you use the command, but do not specify ON or OFF, BTEQ assumes ON and ALL.

If you are going to produce several reports with different FOLDLINE settings during the same BTEQ session, the FOLDLINE option must be explicitly set to OFF before specifying each new FOLDLINE command.
Example 1

The following commands show two select operations from the tables in the Personnel sample database. The width for the report has been purposely set to 40, which is smaller than the length of the rows to be returned by the select statements. Repeating values for the first two returned columns have been suppressed to facilitate reading of the results. The first select is executed with FOLDLINE OFF for all columns. The second select is executed with FOLDLINE set to ON for column 2.

```sql
.SET WIDTH 40;
.SET SUPPRESS ON 1,2;
.SET FOLDLINE OFF;
SELECT
  e.EmpNo, e.Name,
  c.WkEnd (FORMAT 'Mmm-dd'),
  c.Hours, p.Description
FROM
  employee e, project p,
  charges c
WHERE e.EmpNo = c.EmpNo
  AND c.Proj_Id = p.Proj_Id
  AND e.DeptNo = 500
ORDER By 1,3;
.SET FOLDLINE OFF;
.SET FOLDLINE ON 2;
SELECT
  e.EmpNo, e.Name,
  c.WkEnd (FORMAT 'Mmm-dd'),
  c.Hours, p.Description
FROM
  employee e, project p,
  charges c
WHERE e.EmpNo = c.EmpNo
  AND c.Proj_Id = p.Proj_Id
  AND e.DeptNo = 500
ORDER By 1,3;
```

BTEQ Response

The results of the first select statement, generated with FOLDLINE OFF ALL in effect, show how data rows can be truncated if they are longer than the report width. The results of the second select statement, generated with FOLDLINE ON 2 in effect, show how BTEQ will fold a returned row onto multiple lines. In this case, the rows have been split after printing column 2’s value.

```sql
*** Query completed. 12 rows found. 5 columns returned.  
*** Total elapsed time was 1 second.

<table>
<thead>
<tr>
<th>Week</th>
<th>EmpNo</th>
<th>Name</th>
<th>Ending</th>
<th>Hours</th>
<th>Project</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10004</td>
<td>Smith T</td>
<td>Jul-29</td>
<td>53.0</td>
<td>Design W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Nov-18</td>
<td>40.0</td>
<td>Design W</td>
</tr>
<tr>
<td>10010</td>
<td>Reed C</td>
<td>Feb-18</td>
<td>12.5</td>
<td>A/R RECV</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Feb-18</td>
<td>10.0</td>
<td>A/P Paya</td>
</tr>
<tr>
<td>10014</td>
<td>Inglis C</td>
<td>Jan-14</td>
<td>20.0</td>
<td>O/E Onli</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Jan-21</td>
<td>30.5</td>
<td>O/E Data</td>
</tr>
</tbody>
</table>
```
Chapter 5: BTEQ Commands
FOLDLINE

Jan-28  30.0 O/E Data
10015 Omura H  Feb-18  30.5 A/P Paya
            Feb-25  24.0 A/R RECV
10016 Carter J  Jan-14  32.0 Design W
            Feb-25  2.5 Design W
            May-20 32.0 Design W
---------------------------------------------------------------------
*** Query completed. 12 rows found. 5 columns returned.
*** Total elapsed time was 1 second.
EmpNo  Name
----- ------------
Week  Ending Hours  Project Description
------ ----- -------------------------
10004 Smith T  Jul-29 53.0 Design Widget Pwr Supply
          Nov-18 40.0 Design Widget Frame
10010 Reed C  Feb-18 12.5 A/R RECV Online System
          Feb-18 10.0 A/P Payable Online System
10014 Inglis C  Jan-14 20.0 O/E Online System
          Jan-21 30.5 O/E Data Base Design
          Jan-28 30.0 O/E Data Base Design
10015 Omura H  Feb-18 30.5 A/P Payable DB Design
          Feb-25 24.0 A/R RECV Online System
10016 Carter J  Jan-14 32.0 Design Widget Pwr Supply
          Feb-25 2.5 Design Widget Frame
          May-20 32.0 Design Widget Pwr Supply
---------------------------------------------------------------------

Example 2
To omit columns 1 and 2 from one report, and then, in the same BTEQ session, to omit column 3 of the next report, first specify the FOLDLINE command for the first BTEQ report:

.SET FOLDLINE 1, 2

Next, set the FOLDLINE command option to OFF for the next BTEQ report:

.SET FOLDLINE OFF

Then specify the FOLDLINE command for the current report:

.SET FOLDLINE 3

You must turn off the previous FOLDLINE option for any previously folded columns so that they appear correctly on subsequent reports.

Example 3
The following hypothetical results show how BTEQ treats column values solely comprised of space characters when using the FOLDLINE command. The first set of response output was generated with FOLDLINE set to OFF for all columns. The second set was generated with FOLDLINE set to ON for all columns. Note how the values for the 5th row visually seem to be
folded lines for the 4th row. The third set also had FOLDLINE set to ON for all columns but also used UNDERLINE ON 1 to clearly show row boundaries.

Response for FOLDLINE OFF ALL:

<table>
<thead>
<tr>
<th>col1</th>
<th>col2</th>
<th>col3</th>
<th>col4</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNO</td>
<td>NOP</td>
<td>OPQ</td>
<td>PQR</td>
</tr>
<tr>
<td>JKL</td>
<td>KLM</td>
<td>LMN</td>
<td></td>
</tr>
<tr>
<td>GHI</td>
<td>HIJ</td>
<td>IJK</td>
<td></td>
</tr>
<tr>
<td>DEF</td>
<td>EFG</td>
<td>FGH</td>
<td></td>
</tr>
<tr>
<td>ABC</td>
<td>BCD</td>
<td>CDE</td>
<td></td>
</tr>
</tbody>
</table>

Response for FOLDLINE ON ALL:

<table>
<thead>
<tr>
<th>col1</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNO</td>
</tr>
<tr>
<td>NOP</td>
</tr>
<tr>
<td>OPQ</td>
</tr>
<tr>
<td>PQR</td>
</tr>
<tr>
<td>JKL</td>
</tr>
<tr>
<td>KLM</td>
</tr>
<tr>
<td>LMN</td>
</tr>
<tr>
<td>GHI</td>
</tr>
</tbody>
</table>

Response for FOLDLINE ON ALL with UNDERLINE ON 1:

<table>
<thead>
<tr>
<th>col1</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNO</td>
</tr>
<tr>
<td>NOP</td>
</tr>
<tr>
<td>OPQ</td>
</tr>
<tr>
<td>PQR</td>
</tr>
<tr>
<td>JKL</td>
</tr>
<tr>
<td>KLM</td>
</tr>
<tr>
<td>LMN</td>
</tr>
<tr>
<td>GHI</td>
</tr>
</tbody>
</table>
FOOTING

Purpose
Specifies the footer text string that appears at the bottom of each page of a report.

Syntax
```
SET FOOTING 'string'
```
where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>text for the footer. The string can contain a maximum of 253 characters, including line separator characters, to describe up to 10 lines.</td>
</tr>
</tbody>
</table>

Usage Notes
You can use the FOOTING command in a Teradata SQL macro.

Using Special Characters
Do not use special characters within the string because they may be interpreted differently by different output devices. You might have to modify a script that uses special characters if you route your output to another device.

Using Apostrophes
If the string will have an apostrophe (single quote) character, use the second form of the FOOTING command (the one with quotes as delimiters) or double the apostrophe. For example:
```
.SET FOOTING "December's Results"
```
or
```
.SET FOOTING 'December''s Results'
```
If the string will not have an apostrophe, the two forms of the FOOTING command are equivalent.

Formatting Footers
You control footing formats using two types of control characters:
Breaking a Line
You can break a footer line and begin a new line of footer text by inserting a pair of slashes (//) at the desired break point. BTEQ allows up to nine line breaks (10 footer lines maximum) within a FOOTER command.

Separating a Line into Sections
You can separate a footer line into as many as three sections by inserting a pair of solid vertical lines at the desired separation points. BTEQ allows only three sections (two pairs of vertical bar separators) for each line of the footer.

<table>
<thead>
<tr>
<th>Specification</th>
<th>Action</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>a footer with no separator characters</td>
<td>leaves the line as one section</td>
<td>the footer is centered.</td>
</tr>
<tr>
<td>one set of separator characters</td>
<td>separates the line into two sections</td>
<td>the first part of the line is left-justified, and the second part of the line is centered.</td>
</tr>
<tr>
<td>two sets of separator characters</td>
<td>separates the line into three sections</td>
<td>the first part of the line is left-justified, the second part of the line is centered, and the third part of the line is right-justified.</td>
</tr>
</tbody>
</table>

Note: You can also use a pair of split vertical bars as a separator character.

Exceeding the Line Separator Limit
If you specify more than two pairs of vertical bar separator characters, the following occurs:

<table>
<thead>
<tr>
<th>System Type</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>channel-attached</td>
<td>BTEQ interprets the additional characters as footer text, and issues a warning that you have exceeded the limit of two pairs of vertical bar separator characters.</td>
</tr>
<tr>
<td>network-attached</td>
<td>BTEQ truncates the additional separators from the footer text.</td>
</tr>
</tbody>
</table>

For example, in a channel-attached system the following command:

```
.SET FOOTING '&DATE||Confidential Report||Part Two||Page&PAGE'
```

produces the warning message:
*** Warning: Maximum of 2 '||' separators is allowed, others ignored.

and produces the following undesirable footing:

90/08/25 Confidential Report Part Two||Page2

**Using Substitution Values**

As shown in the preceding example, you can also use the following substitution values anywhere in a footer string:

- &DATE
- &TIME
- &PAGE
- &n

where:

<table>
<thead>
<tr>
<th>Substitution Value</th>
<th>Provides</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;DATE</td>
<td>current date in YY/MM/DD format</td>
</tr>
<tr>
<td>&amp;TIME</td>
<td>current time in HH:MM format</td>
</tr>
<tr>
<td>&amp;PAGE</td>
<td>current page number. (If you specify a footer and do not use the &amp;PAGE substitution value, the page number will not appear in the footer.)</td>
</tr>
<tr>
<td>&amp;n</td>
<td>most recent value in the nth expression of the selected row that immediately precedes the footing. You cannot use the &amp;n substitution value in the following cases:</td>
</tr>
<tr>
<td></td>
<td>- When you specify .SET FOLDLINE ON m if the &amp;n value is greater than the value of m, and &amp;n is greater than one.</td>
</tr>
<tr>
<td></td>
<td>- When you specify .SET FOLDLINE ON ALL.</td>
</tr>
<tr>
<td></td>
<td>For non-EXPORTed output within z/OS BTEQ Unicode sessions, this value must only be comprised of LATIN1 characters.</td>
</tr>
</tbody>
</table>

**Specifying a Footer That Is Wider than Page**

If you specify a footer text string that is wider than the page width, BTEQ first truncates the middle portion of the footer. When that is gone, if more must be truncated, BTEQ truncates the right and left portions until the footer fits on the page. Truncation is from the right, as shown in the following example:

```sql
.set format on
.set footing 'This is an example of a footing that is too long for the page and is truncated starting from the right'
```

A subsequent SQL SELECT statement would return the following footer:

This is an example of a footing that is too long for the page and 1
If You Do Not Specify a Footer

If you do not specify a footer, BTEQ uses the lines of the footer area at the bottom of the page for additional rows of the results from the SQL SELECT statement.

**Note:** If you set the FORMAT command option to OFF, BTEQ ignores the FOOTING command when formatting output, even though you can see the footing specifications in the response to a SHOW CONTROLS command. Always set the FORMAT command option to ON if you want a footer to appear in your report.

**Example 1**

**BTEQ Commands:**

```plaintext
database personnel;
.set defaults on
.set format on
.set footing 'Confidential Report'
select * from department
;
```

**BTEQ Response**

In response to the Example 1 commands, BTEQ returns:

```
*** Query completed. 5 rows found. 4 columns returned.
90/07/25  select * from department;  Page 1
DeptNo  DeptName  Loc  MgrNo
------  --------------  ---  -----
500  Engineering  ATL  10012
700  Marketing  NYC  10021
300  Exec Office  NYC  10018
600  Manufacturing  CHI  10007
100  Administration  NYC  10011
Confidential Report
```

**Example 2**

**BTEQ commands:**

```plaintext
database personnel;
.set defaults
.set format on
.footing "This is an example of a continued footing with incorrect spacing"
select * from department
;
```

**BTEQ Response**

In response to the Example 2 commands, BTEQ returns:

```
*** Query completed. 5 rows found. 4 columns returned.
90/07/25  select * from department;  Page 1
DeptNo  DeptName  Loc  MgrNo
------  --------------  ---  -----
500  Engineering  ATL  10012
700  Marketing  NYC  10021
300  Exec Office  NYC  10018
```
Chapter 5: BTEQ Commands

FOOTING

This is an example of a continued footing with incorrect spacing.

Example 3

BTEQ Commands:

database personnel;
.set defaults
.set format on
.footing '&date||Confidential||Page&Page'
select * from department
;

BTEQ Response

In response to the Example 3 commands, BTEQ returns:

*** Query completed. 5 rows found. 4 columns returned.
90/07/25          select * from department;          Page 1
DeptNo  DeptName  Loc  MgrNo
------  --------------  ---  -----  
 500  Engineering  ATL  10012
 700  Marketing  NYC  10021
 300  Exec Office  NYC  10018
 600  Manufacturing  CHI  10007
 100  Administration  NYC  10011

90/07/25         Confidential          Page 1

Example 4

The Example 4 commands in a Teradata SQL macro appear as:

ECHO '.SET FOOTING ''&DATE||Confidential||Page&PAGE'' '

or as:

ECHO '.SET FOOTING "&DATE||Confidential||Page&PAGE"';
**Purpose**
Enables all of the page-oriented formatting commands, or disables them and centers the response from SQL SELECT statements, using the value of the WIDTH command option to determine the space available.

**Syntax**
```
SET FORMAT [ON | OFF]
```

**Note:** For z/OS, the value of FORMAT is OFF by default for interactive mode, and ON for batch mode.

**Usage Notes**
In response to the FORMAT command, BTEQ either recognizes or ignores the format specifications that have been set using the following page-oriented formatting commands:

- FOOTING
- FORMCHAR
- HEADING
- RTITLE
- PAGEBREAK
- PAGELENGTH

When you set the FORMAT command option to OFF, BTEQ ignores the configuration of these command options, even though they are shown as ON in the response to a SHOW CONTROLS command.

In general, setting FORMAT to OFF is appropriate for directing the output to the screen, and ON is appropriate for directing the output to a file for printing.

When BTEQ runs in any mode other than z/OS batch, the default is OFF because page formatting does not apply to the terminal screen when BTEQ is used online. In batch mode, the default is ON because results are printed according to page formats.

If a FORMAT command is not specified, the value is OFF by default. If you use the command and do not specify ON or OFF, BTEQ assumes ON.

You can use the FORMAT command in a Teradata SQL macro.

**Example 1**
The following example shows the effects of setting the FORMAT command option to ON:
```
database personnel;
```
Chapter 5: BTEQ Commands

FORMAT

.set defaults
.set format on
.set heading 'Format on'
.set sidetitles on
.set skipline on 1
.set width 40
.set pagelength 66
.footing 'Super Confidential Report'

select MgrNo(title ''), DeptName(title ''), Loc(title ''), DeptNo()
from department
order by MgrNo;

BTEQ Response

In response to the Example 1 commands, BTEQ returns:

*** Query completed. 5 rows found. 4 columns returned.
Format on
10007  Manufacturing  CHI  600
10011  Administration  NYC  100
10012  Engineering  ATL  500
10018  Exec Office  NYC  300
10021  Marketing  NYC  700
Super Confidential Report

Example 2

The Example 2 command in a Teradata SQL macro appears as:

ECHO '.SET FORMAT ON';

Example 3

The following example shows the effects of setting the FORMAT command option to OFF:

database personnel;
.defaults
.format off
.heading 'Format off'
.set sidetitles on
.set skipline off
.set width 40
.set pagelength 66
.footing 'Super duper Confidential Report' = 1
;

BTEQ Response

In response to the Example 3 commands, BTEQ returns:

*** Query completed. 5 rows found. 4 columns returned.
10007  Manufacturing  CHI  600
10011  Administration  NYC  100
10012  Engineering  ATL  500
10018  Exec Office  NYC  300
10021  Marketing  NYC  700
FORMCHAR

Purpose
Provides format control for printing reports. The command has four options: ON, OFF, DEFAULT or any HEX sequence that a printer translates into a printer escape sequence (for example, "0C"XB or 'oc'xb).

Syntax

\[
\text{SET FORMCHAR} [\text{OFF} | \text{ON} | \text{"HEX sequence ' xb} | \text{"HEX sequence ' xb} | \text{DEFAULT}]
\]

where:
- OFF = initial default for network-attached systems
- DEFAULT = initial default for channel-attached systems

Usage Notes
Use the FORMCHAR command when you need to specify form feed characters for printing reports.

For the FORMCHAR command to be effective, you must first set the FORMAT command option to ON. For example, with the FORMCHAR command option set to ON, BTEQ sends a form feed character to the mainframe printer and puts “1” in column one.

Certain printers accept only a special numeric sequence in HEX to produce a form feed at the end of a page. You can use either uppercase or lowercase characters when entering the HEX sequence. If used, the HEX sequence must be less than or equal to ten HEX characters. Be careful when choosing this option, because character sets are implementation-dependent.

If no option is entered, or if an invalid option is entered, the value of FORMCHAR is reset to its initial default value.

The following commands are affected by the FORMCHAR command:

- SHOW CONTROLS
- HELP BTEQ
- FORMAT

You can use the FORMCHAR command in a Teradata SQL macro.
Example 1
To specify form feed ON for a network-attached system, type:

```
.SET FORMAT ON
.SET FORMCHAR ON
```

Example 2
Specify the command for HEX as follows:

```
.SET FORMAT ON
.SET FORMCHAR "0C"xb
```

Example 3
The Example 2 command in a Teradata SQL macro appears as:

```
ECHO '.SET FORMCHAR "0C"xb';
```

Example 4
To enter the default for channel-attached operating systems, type:

```
.help bteq
.show controls
.set formchar 'e7'xb
.show controls
.set formchar default
.show controls
```
FULLYEAR

Purpose
Enables you to use a four-digit year format (YYYY) to comply with the year 2000 format. The default is a two-digit year format (YY).

Syntax

```
.SET FULLYEAR [ON | OFF]
```

Usage Notes
If the FULLYEAR command is used without specifying ON or OFF, BTEQ functions as if the command is set to ON.

FULLYEAR only affects the date in the header, footer, and title lines. The date data returned from the SQL is not affected by the FULLYEAR command.

If the FULLYEAR command has not been used, its value is ON by default.

Example 1
To enable a four-digit year format to comply with the year 2000 format, type:

```
.SET FULLYEAR ON
```

Example 2
The Example 1 command in a Teradata SQL macro appears as:

```
ECHO '.SET FULLYEAR ON';
```
GOTO

Purpose
Skips over all intervening BTEQ commands and SQL statements until a specified label is encountered, then resumes processing in sequence.

Syntax
\[ .GOTO \textit{labelname} \]

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\textit{labelname}</td>
<td>name of the statement to which control transfers, as specified by a subsequent BTEQ LABEL command. For more information, see “LABEL” on page 200.</td>
</tr>
</tbody>
</table>

Usage Notes
The GOTO command only transfers control forward; it cannot transfer control to a statement that occurs before the GOTO command. If you do not provide a LABEL command statement specifying a label that matches the requested label, BTEQ terminates the script.

The GOTO command is frequently used with the IF...THEN... command.

Example
The following example uses GOTO commands to transfer control forward to the LABEL statements SECTIONA or SECTIONB:

```sql
database personnel;
select empno ,hours from charges
where proj_id = 'oe1-0001'
order by empno;
.IF activitycount = 0 then .GOTO SECTIONA
.IF activitycount >= 1 then .GOTO SECTIONB
.label SECTIONA
.REMARK 'Zero Hours on Account'
.label SECTIONB
.REMARK 'Total Hours on Account'
.logoff
.exit
```
### HANG

**Purpose**
Pauses BTEQ for a specified period of time.

**Syntax**

```
.HANG n
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>n</code></td>
<td>amount of time, in seconds, that you want to pause the BTEQ job. The range of <code>n</code> is 0 to 2147483647. The value is 360 seconds (6 minutes) by default.</td>
</tr>
</tbody>
</table>

**Usage Notes**
Use the HANG command to specify a period of time to pause a program.

You can use the HANG command in a Teradata SQL macro.

If no value is entered, BTEQ will pause for the default six minutes.

**Example 1**
To specify a half-minute pause, type:

```
.HANG 30
```

**Example 2**
The Example 1 command in a Teradata SQL macro appears as:

```
ECHO '.HANG 30';
```
HEADING

Purpose
Specifies a header that appears at the top of each page of a report.

Syntax
```
SET HEADING "string"
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>text for the header. The string can contain a maximum of 253 characters, including line separator characters, to describe up to 10 lines.</td>
</tr>
</tbody>
</table>

Usage Notes
The HEADING command and the RTITLE command are alternate versions of the same command. They both specify the top ten lines on each page of a report. If you use both commands, the most recent one overrides the earlier one.

The RTITLE command automatically provides the page number and the date; the HEADING command does not. When using the HEADING command, you must provide the &PAGE and &DATE substitution values in you header string if you want the page numbers and date to appear in your header.

You should not use special characters within the header string because they may be interpreted differently by different output devices. You might have to modify a script that uses special characters if you route your output to another device.

If the string will have an apostrophe (single quote) character, use the second form of the HEADING command (the one with double quotes as delimiters) or double the apostrophe. For example,

```
.SET HEADING "December's Results"
```

or

```
.SET HEADING 'December''s Results'
```

If the string will not have an apostrophe, the two forms of the HEADING command are equivalent.
**Formatting Headings**

You control heading formats using two types of control characters:

<table>
<thead>
<tr>
<th>Control Characters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>double slashes (/)</td>
<td>breaks the line, and begins a new line</td>
</tr>
<tr>
<td>solid vertical bars (</td>
<td></td>
</tr>
</tbody>
</table>

**Breaking a Line**

You can break a header line and begin a new line of header text by inserting a pair of slashes (/\) at the desired break point. BTEQ allows up to nine line breaks (10 header lines maximum) within a HEADING command.

**Separating a Line into Sections**

You can separate a header line into as many as three sections by inserting a pair of solid vertical lines at the desired separation points. BTEQ allows only three sections (two pairs of vertical bar separators) for each line of the header.

<table>
<thead>
<tr>
<th>Specification</th>
<th>Action</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>a header with no separator characters</td>
<td>leaves the line as one section</td>
<td>the header is centered.</td>
</tr>
<tr>
<td>one set of separator characters</td>
<td>separates the line into two sections</td>
<td>the first part of the line is left-justified, and the second part of the line is centered.</td>
</tr>
<tr>
<td>two sets of separator characters</td>
<td>separates the line into three sections</td>
<td>the first part of the line is left-justified, the second part of the line is centered, and the third part of the line is right-justified.</td>
</tr>
</tbody>
</table>

**Note:** You can also use a pair of split vertical bars as a separator character.

If you specify more than two pairs of vertical bar separator characters, BTEQ does the following:

<table>
<thead>
<tr>
<th>System Type</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>channel-attached</td>
<td>BTEQ interprets the additional characters as header text, and issues a warning that the limit of two pairs of vertical bar separator characters has been exceeded.</td>
</tr>
<tr>
<td>network-attached</td>
<td>BTEQ truncates any additional text including the extra (</td>
</tr>
</tbody>
</table>
Using Substitution Values
You can use the following substitution values anywhere in a header string:

- &DATE
- &TIME
- &PAGE
- &n

where:

<table>
<thead>
<tr>
<th>Substitution Value</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;DATE</td>
<td>current date in YY/MM/DD format</td>
</tr>
<tr>
<td>&amp;TIME</td>
<td>current time in HH:MM format</td>
</tr>
<tr>
<td>&amp;PAGE</td>
<td>current page number. (If you specify a header and do not use the &amp;PAGE substitution value, the page number will not appear in the header.)</td>
</tr>
<tr>
<td>&amp;n</td>
<td>value in the nth expression of the selected row that immediately follows the header.</td>
</tr>
</tbody>
</table>

The &n substitution value can be used in the following cases:
- when you specify .SET FOLDLINE ON m.
- when &n is greater than the value of m, and &n is greater than one
- when you specify .SET FOLDLINE ON ALL.

For non-EXPORTed output within z/OS BTEQ Unicode sessions, this value must only be comprised of LATIN1 characters.

Specifying a Header That Is Wider Than the Page
If you specify a header string that is wider than the page, BTEQ truncates from the right until the header fits on the page. (You can continue a header from one line to the next by including a dash character (-) as the last nonblank character of the line to be continued.)

If you do not use this command to specify a heading, or the RTITLE command to specify a title, the default one-line header/title provides the date, as many characters of the SQL SELECT statement that fit, and the page number, formatted as follows:

&DATE||as many characters of SELECT as will fit||Page&PAGE

Note: The SELECT statement is the statement for which BTEQ is generating a report. The titles only appear in response to a SELECT statement.

Note: If you set the FORMAT command option to OFF, BTEQ ignores the HEADING command when formatting output, even though you can see the heading specifications in the response to a SHOW CONTROLS command. Always set the FORMAT command option to ON if you want a header to appear in your reports.

When using the HEADING command with the FOLDLINE command, you can show the heading on the first page.
You can use the HEADING command in a Teradata SQL macro.

**Example 1 1**
Example BTEQ commands and response follow.

**BTEQ Commands**
```
database personnel;
.set defaults on
.set format on
.set heading 'Confidential Report'
.omit on
select * from employee
;
```

**BTEQ Response**
```
*** Query completed. 21 rows found. 12 columns returned.
Confidential Report
```

**Example 1 2**
Example BTEQ commands and response follow.

**BTEQ Commands**
```
database personnel;
.set defaults
.set format on
.set heading 'This is an example -
of a continued -
heading'
.omit on
select * from employee
;
```

**BTEQ Response**
```
*** Query completed. 21 rows found. 12 columns returned.
This is an example of a continued heading
```

**Example 1 3**
Example BTEQ commands and response follow.

**BTEQ Commands**
```
database personnel;
.set defaults
.set format on
.heading '&date||Marketing||Page&Page//Third Quarter//Chicago'
.omit on select * from employee
;
.set format off
.logoff
.exit
```

**BTEQ Response**
```
*** Query completed. 21 rows found. 12 columns returned.
```
Example 1 4
The command in a Teradata SQL macro appears as:

```sql
ECHO '.SET HEADING
'.'&date||Marketing||Page&Page//'Third -
Quarter//'Chicago'' ';
```

or as:

```sql
ECHO '.SET HEADING
&date||Marketing||Page&Page//' Third -
Quarter//'Chicago'';
```
**HELP BTEQ**

**Purpose**
Returns a list of all available BTEQ commands and non-graphic representations of their syntax. (The SQL Online Help feature provides similar help for SQL commands.)

**Syntax**
```
HELP BTEQ
```

**Usage Notes**
The HELP BTEQ command provides accurate spelling and syntax for all BTEQ commands. The HELP BTEQ command is not suppressed by the QUIET command.

The HELP BTEQ display uses the following characters to represent the command syntax in a non-graphic format:
- Brackets, [ ] indicate optional choices that may be omitted, entirely.
- Braces, { } indicate required choices, one of which must be stated.
- A vertical bar | separates different values for options.
- Uppercase characters represent reserved words that must be specified exactly as shown.
- Lowercase characters represent variables that you must provide.

You can use the HELP BTEQ command in a Teradata SQL macro.

**Example 1**
To request help, type:
```
HELP BTEQ
```

**BTEQ Response**
BTEQ is a general-purpose utility which enables you to submit Teradata SQL queries to the Teradata Database; it formats the results and returns them to the screen, file, or printer.

This statement is then followed by an alphabetical listing of the BTEQ commands.

**Example 2**
The Example 1 command in a Teradata SQL macro appears as:
```
ECHO '.HELP BTEQ';
```
Purpose
From channel-attached systems, the HX (Halt eXecution) command aborts the previous request and any transaction in which the request was embedded, logs off the current set of sessions, and exits from BTEQ. (The HX command is not valid from network-attached client systems.)

Syntax

```
HX
```

Usage Notes
Before you can halt execution, BTEQ must first be interrupted. See “Using the Break Key on z/OS” on page 82 for instructions.

The results depend on the timing of the abort request:

<table>
<thead>
<tr>
<th>Attempted abort reaches Teradata Database</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>before the response to the original request has been sent,</td>
<td>you receive a failure message because the abort causes the original request to fail. The response to the original request is discarded, and any transaction in which the original request was embedded is aborted. In this case, the abort operation backs out any changes that the previous request made to the database.</td>
</tr>
<tr>
<td>after the response to the original request has been sent,</td>
<td>the attempted abort is ignored and the original request takes effect. If the original request was embedded in a transaction, the entire transaction is aborted and any effect that the statements in the transaction had on the database are backed out. In either case, the current set of sessions is logged off, and you exit from BTEQ.</td>
</tr>
</tbody>
</table>
**Purpose**

Tests the validity of the condition stated in the IF clause.

<table>
<thead>
<tr>
<th>Condition Status</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>BTEQ executes the action stated in the THEN clause.</td>
</tr>
<tr>
<td>False</td>
<td>BTEQ ignores the action in the THEN clause and proceeds to the next command statement.</td>
</tr>
</tbody>
</table>

**Syntax**

```
.bteq_command or sql_request
IF ERRORCODE operator n THEN bteq_command sql_request
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bteq_command</code> or <code>sql_request</code></td>
<td>BTEQ command or Teradata SQL statement that you want to execute when the IF condition is true.</td>
</tr>
</tbody>
</table>
| `n` | a number in the range of:
| | • 0 to 9000, inclusive, if you specify the ERRORCODE option
| | • 0 to 4,294,967,295 if you specify the ACTIVITYCOUNT option
| | Or, if you specify the ERRORLEVEL option, `n` is an assigned error level number that can be 0, 2, 4, 6, 8, or higher. |
| `operator` | one of the following comparison operators:
| | = (equal to)
| | > (greater than)
| | < (less than)
| | >= (greater than or equal to)
| | <= (less than or equal to)
| | <> (not equal)
| | != (not equal)
| | ~= (not equal)
| | ^= (not equal) |
Usage Notes

<table>
<thead>
<tr>
<th>Specified Option</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERRORCODE</td>
<td>BTEQ substitutes for ERRORCODE the actual value of the error code that was generated by the last Teradata SQL request.</td>
</tr>
<tr>
<td></td>
<td>If the last request completed successfully, the value of ERRORCODE is zero.</td>
</tr>
<tr>
<td>ERRORLEVEL</td>
<td>BTEQ returns the severity level code associated with that error level.</td>
</tr>
<tr>
<td>ACTIVITYCOUNT</td>
<td>BTEQ substitutes for ACTIVITYCOUNT the actual value of the number of rows affected by the last Teradata SQL statement. For a request resulting in a selection of rows, this will be the number of rows actually returned to BTEQ from the database.</td>
</tr>
</tbody>
</table>

You can use the IF...THEN... command in a Teradata SQL macro, if the THEN clause provides a valid BTEQ command that is allowed in a Teradata SQL macro.

**Example 1**
To transfer control to NXTREPORT if the error code generated by the last Teradata SQL request is 3000, type:

```
.IF ERRORCODE = 3000 THEN .GOTO NXTREPORT
```

**Example 2**
To set a return code to 1 if a Teradata SQL request finds no rows, type:

```
.IF ACTIVITYCOUNT = 0 THEN .QUIT 1
```

**Example 3**
When used in a Teradata SQL macro, the IF...THEN... command has the format:

```
ECHO 'IF ERRORCODE = 3000 THEN .REMARK ''Next'' ';
```
**IMPORT**

**Purpose**

Opens a channel- or network-attached system file, of the specified format, to provide data for USING modifiers of subsequent SQL statements.

**Syntax**

For Channel-Attached Systems:

```
IMPORT
  DDNAME ddname, SKIP = n
  DATA
  INDICDATA
  VARTEXT
    'i'
    'c'

where:
```

For Network-Attached Systems:

```
IMPORT
  REPORT
    INDICDATA
      LOBCOLS = n
      TOTCOLS = n
    DATA
    VARTEXT
      'i'
      'c'

FILE filename, SKIP = n

AXSMOD
  modname
  'nit_string'
```

where:
## Syntax Element

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>default field delimiter character is a broken vertical bar whose ASCII equivalent is 0x7C, TD_EBCDIC equivalent is 0x6A, UTF8 equivalent is 0x7C and UTF16(BE) equivalent is 0x00 0x7C.</td>
</tr>
<tr>
<td>AXSMOD</td>
<td>indicates that data is to be imported using an access module. The file size can be greater than 2 GB.</td>
</tr>
<tr>
<td>c</td>
<td>optional field delimiter character (must be enclosed in single quotation marks).</td>
</tr>
<tr>
<td>DATA, INDICDATA, or REPORT</td>
<td>format of the data to be read by BTEQ.</td>
</tr>
<tr>
<td></td>
<td>If DATA or INDICDATA is specified, BTEQ expects the incoming data to be in FastLoad format, such as that produced by a BTEQ EXPORT DATA or BTEQ EXPORT INDICDATA command, or by FastExport.</td>
</tr>
<tr>
<td></td>
<td>If the REPORT format is specified, BTEQ expects the incoming data to be in the format produced by a BTEQ EXPORT REPORT command.</td>
</tr>
<tr>
<td></td>
<td>For z/OS, the lines in the import file are truncated to 254 characters by default. The width limit may be increased to 65531 characters using the WIDTH command.</td>
</tr>
<tr>
<td></td>
<td>The REPORTWIDE keyword is also accepted. This keyword specifies that the REPORT format for the data should be used, and changes the WIDTH setting to 32765. Since this keyword is deprecated, a SET WIDTH command should be used instead.</td>
</tr>
<tr>
<td></td>
<td>See the WIDTH command “WIDTH” on page 322.</td>
</tr>
<tr>
<td>ddname</td>
<td>name of the z/OS ICL DD statement that defines the channel-attached system file from which BTEQ reads data.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Though you can use the keyword FILE instead of the keyword DDNAME for channel-attached systems, be very careful when you do because the implementation is for the concept DDNAME, not the concept FILE.</td>
</tr>
<tr>
<td>filename</td>
<td>name of the network-attached system file from which BTEQ reads data.</td>
</tr>
<tr>
<td></td>
<td>If the name of the file includes a comma, semicolon, or space character, enclose the entire file name in either single or double quotation marks.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> File names are case-sensitive on systems running under UNIX, and they are not case-sensitive on systems running under Windows. Whenever other command options are specified following the file name in the Windows environment, BTEQ requires the file name to be enclosed by either single or double quotation marks.</td>
</tr>
<tr>
<td></td>
<td>Though you can use the keyword DDNAME instead of the keyword FILE for network-attached systems, be very careful when you do because the implementation is for the concept FILE, not the concept DDNAME.</td>
</tr>
<tr>
<td>'init-string'</td>
<td>name of the initiation string used to start the import process.</td>
</tr>
<tr>
<td>LOBCOLS</td>
<td>number of LOB values which need to be elicited separately from other row data.</td>
</tr>
<tr>
<td>modname</td>
<td>name of the file that is to receive the imported data.</td>
</tr>
</tbody>
</table>
For channel-attached systems, BTEQ supports all sequential files with fixed, variable, or undefined length record formats.

When the record length of data used during an import operation is greater than 252 bytes, BTEQ displays the following message, indicating that the buffer size has been increased:

Growing buffer to 32000

The buffer size is actually increased to 32006, where the last six bytes are used for the parcel flavor (two bytes) and parcel length (four bytes).

The DATA and INDICDATA keywords specify the format of the data to be written to Teradata Database.

DATA format is used in Record Mode, which does not send explicit null indicator bits.

INDICDATA format is used in Indicator Mode, which does send explicit null indicator bits.

For the character sets KANJISJIS_0S, KANJIEUC_0U, TCHBIG5_1R0, and SDTCHBIG5_3R0, the default is Record (DATA) Mode, not Field Mode.

You cannot use the IMPORT command in a Teradata SQL macro.

**Note:** It is the user’s responsibility to correctly describe the input data in the USING modifier when utilizing the IMPORT command. This general IMPORT command warning applies to all IMPORT types.

**Note:** Use the SHOW CONTROLS command to find out the current record format of the IMPORT command.

**Note:** For Unicode sessions, the encoding of import files must match the encoding of the session character set.

### DATA and INDICDATA Keywords

**Network-Attached Systems**

If you use the keyword DATA or INDICDATA, each record must be in the following format:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>n</em></td>
<td>number of lines skipped from the beginning of the file before reading data. The value is zero (0) by default.</td>
</tr>
<tr>
<td>TOTCOLS</td>
<td>INDICDATA form import data records and required for AS DEFERRED transfer of LOBs.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Mainframe BTEQ does not support deferred mode transfer of LOBs. Workstation BTEQ does not support deferred mode transfer of LOBs using Unicode session charsets.</td>
</tr>
<tr>
<td>VARTEXT</td>
<td>record format as variable length character fields, with each field separated by a delimiter character.</td>
</tr>
</tbody>
</table>
Chapter 5: BTEQ Commands

IMPORT

The record begins with a two-byte length field, that contains an unsigned binary integer whose value is the number of bytes in the record (not including the bytes in the length field and not including the byte(s) in the end-of-record field).

- The record follows with binary-formatted fields that contain the column values of the selected row in record or indicator format.
- The record ends with an end-of-record (line) indicator, appropriate for the workstation and operating system. For example, for UNIX, the indicator is a new line. For Windows, the indicator is a carriage return/line feed pair. In all cases, the end-of-record character, or characters, are in the bit-format appropriate for the workstation. For example, in ASCII for an ASCII workstation.

**Caution:** Numeric data fields will be corrupted if the data is reimported from an EXPORT REPORT format.

When using a file originally exported using EXPORT command of BTEQ as the source for IMPORT command across a different platform type, ensure that the endianness type of both platforms is the same. This can be verified from the “Client Platform Byte Order” tab in the output of SHOW CONTROLS command.

**Channel-Attached Systems**

If you use the keyword DATA or INDICDATA, each record must consist of binary-formatted fields that contain the column values of the selected row in record or indicator format.

**Note:** For both network-attached and channel-attached systems a BOM is not allowed at the beginning of a DATA (or INDICDATA) import file when a Unicode session character set is being used.

**LOBCOLS and TOTCOLS Keywords**

The network-attached system-only LOBCOLS keyword indicates the number of LOB values which need to be elicited separately from other row data through ElicitData protocol as indicated by the AS DEFERRED phrase within the request’s USING modifier clause.

The network-attached system-only TOTCOLS keyword is valid only for INDICDATA form import data records, and required for AS DEFERRED transfer of LOBs. The TOTCOLS keyword is used to specify the total number of columns referred to in the request’s USING modifier clause.

All LOBs to be transferred AS DEFERRED must be represented by VARCHAR filename values that exist together as the first portion of an import data record. The maximum size for the filename containing a LOB value is 1024 bytes.

BTEQ replaces the VARCHAR filename values with 4-byte integer tokens before submitting the request to the database. When a single-byte filename is supplied, three bytes (two for the length of the VARCHAR value and one for the filename value) get replaced by four bytes. This replacement means that the overall size of the import data record may become too large to successfully contain all generated tokens.

The maximum value that can be supplied for the LOBCOLS option or the TOTCOLS option is based on the maximum number of data values that Teradata Database allows to be described by any requests’ USING modifier.
Note: The use of Unicode sessions to transfer LOBs in deferred mode is not supported.

Multiple sessions can be used to take advantage of the database’s ability to handle requests in parallel. However, each session used will require its own dedicated file handle.

BTEQ allows for up to 200 sessions to be used. The OS platform in use for the BTEQ process may not allow 200 files to be opened simultaneously. If BTEQ is not able to open the file, the associated request will be aborted. Therefore, the appropriate number of actual sessions to use should be determined in advance of running LOB import jobs for production environment purposes.

REPORT Keyword

If the keyword REPORT is used, each record must be in the format generated by the REPORT keyword of the BTEQ EXPORT command, except that certain records must be removed (for instance, by an editor).

Caution: In REPORT format, BTEQ uses Field Mode to return all data values, including numeric data, in character format. Undesirable results may be obtained with numeric data fields when data is reimported that was exported in REPORT format.

Note: The REPORT parameter is not supported under the Kanji character sets KANJISJIS_0S or KANJIEUC_0U, or the Chinese character sets TCHBIG5_1R0 or SDTCHBIG5_3R0.

Note: On network-attached systems, a BOM is optional at the beginning of a UTF8 or UTF16 REPORT import file when a Unicode session character set is being used.

VARTEXT Keyword

BTEQ supports VARTEXT records by converting each delimited data item in the input record into a VARCHAR data item. Any empty data item causes the data item’s corresponding field to be nulled.

BTEQ interprets the following as empty data items:

- Two adjacent delimiters.
- A delimiter as the very first character of the record.
- A delimiter as the very last character of the record.

You can specify a character other than the default “|” as your delimiter. The following rules for specifying a valid optional character apply:

- No control character other than TAB can be a delimiter.
- Any character that appears in the data cannot be a delimiter.
- Delimiters can only be a single-character sequence.
- For Workstation BTEQ, the delimiter can be a multi-byte character for Unicode sessions only.
- For Mainframe BTEQ, the delimiter must always be a single-byte character.

Rules for all Platforms

- The only acceptable data types for VARTEXT records are VARCHAR, VARBYTE, and LONG VARCHAR. Undesirable results will occur if other data types are used.
Because VARTEXT files are opened in text mode, unexpected results may occur if VARTEXT data contains embedded control characters such as null (0x00), linefeed (0x0A) or sub (0x1A). Control characters are accepted for DATA/INDICDATA imports.

The number of data items in the input record must be equal to the number of fields defined in the USING clause.

Trailing delimiters are never ignored. A delimiter at the end of the input record indicates an ending null value is generated by BTEQ. The null must have a corresponding field defined in the USING clause. If the USING field is not defined, the import may fail because the number of indicator bit bytes sent does not match the number the DBS expects, which is indicated by the USING clause. BTEQ relies on the DBS to detect this problem. The failure, typically, is indicated by an 2673 error with a message stating the source parcel length does not match the data defined. There are two alternatives ways to solve this problem so the expected number of indicator bytes matches what BTEQ sends. The first alternative is to not use a trailing delimiter. The second is to add a dummy last column to the USING clause.

Two consecutive delimiter characters specify that the corresponding field should be nulled. If the record starts with a delimiter, the first field will be nulled.

There should always be one less delimiter than the total number of columns specified in the USING clause.

Note: A BOM is optional at the beginning of a UTF8 or UTF16 VARTEXT import file when a Unicode session character set is being used.

Rules for Workstation Platform

Each record must end with an end-of-record (line) indicator, appropriate for the workstation and operating system. For example, for UNIX, the indicator must be a line feed character. For Windows, the indicator must be a carriage return/line feed pair. In all cases, the end-of-record character, or characters, must be in the bit-format appropriate for the workstation.

Rules for Mainframe Platform

If you use a space/blank character as delimiter, no empty trailing fields will be allowed in the record. If one or more fields are null at the end of a record, and the delimiter character is a space/blank, the format of the record will not be preserved. The trailing nulls and blanks will be truncated, resulting in a DBS error when the record is inserted into a table.

If you want trailing spaces/blanks to be part of the data, end the record with a delimiter.

When using a Unicode session charset, the delimiter character employed in VARTEXT import Unicode data files must be a Unicode code point equivalent to the delimiter character employed for the associated EBCDIC import statements within the z/OS BTEQ script. There are certain EBCDIC characters that will need special consideration to ensure the code points match.

For example, EBCDIC has two vertical bar characters: a solid bar (0x4F) and a broken bar (0x6A).

The EBCDIC solid bar character maps to 0x008D in UTF16 and is called a vertical line.

The EBCDIC broken bar character maps to 0x007C in UTF16 and is called the broken bar.
SET REPEATSTOP ON can be used to stop inserting if an error is encountered during processing of a VARTEXT record. By default it rejects the record and continues.

**Opening and Closing Import Files**

BTEQ opens the import file in response to an IMPORT command. BTEQ closes the import file whenever one of the following conditions occurs:

- BTEQ encounters an end-of-file condition.
- You use another IMPORT command.
- You use a LOGOFF (EXIT/QUIT) command.

**Note:** For information on I/O errors and abends, refer to “I/O Errors and Abends” on page 92.

**Example 1**

To specify an input file with the ddname of NEWDAT, type:

```
.import data ddname=NEWDAT
```

**Example 2**

Always avoid multiple USING clauses with the BTEQ IMPORT command whenever the REPEAT or = command can be used instead. Multiple USING clauses can be useful, however, when inserting or updating data in different tables if you are using the same data form as the import data file. For example:

```
.import data file = test.data;
using (c1 integer, c2 integer)
insert into table1 (c1, c2)
values (:c1, :c2);

using (c1 integer, c2 integer)
insert into table2 (c1, c3)
values (:c1, :c2);

using (c1 integer)
update table3 set c1 = :c1 where c2 = :c1;
```

where:

<table>
<thead>
<tr>
<th>Table</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>table1</td>
<td>'ct table1 (c1 integer, c2 integer)'</td>
</tr>
<tr>
<td>table2</td>
<td>'ct table1 (c1 integer, c2 char(1), c3 integer)'</td>
</tr>
<tr>
<td>table3</td>
<td>'ct table1 (c1 integer, c2 integer, c3 integer)'</td>
</tr>
</tbody>
</table>

**Example - AXSMOD**

The following example imports data from a file named EXPORT.DAT; the data imported from this file can be more than 2 GB:

```
.import data file = 'EXPORT.DAT' AXSMOD
```
INDICDATA

**Purpose**
Specifies the mode of information returned from Teradata Database in response to SQL SELECT statements.

**Syntax**

```plaintext
SET INDICDATA OFF
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFF</td>
<td>Field Mode.</td>
</tr>
<tr>
<td>ON</td>
<td>Indicator Mode.</td>
</tr>
</tbody>
</table>

**Note**: The BTEQ INDICDATA command does not affect the data returned from Teradata Database. It only specifies the format of the data.

**Usage Notes**

By default, BTEQ returns data in Field Mode, and formats it according to accompanying FORMAT commands. In Field Mode, all data values are returned in character format. When the INDICDATA command option is set to ON, data is returned in Indicator Mode. In Indicator Mode, BTEQ does not format data. Instead, each selected row is presented in the format specified in the Teradata SQL SELECT statement.

Each row of data returned by Teradata Database in Indicator Mode begins with the indicator variables for the data values in that row. One indicator bit corresponds to each data item, indicating whether or not a value represents a null as follows:

- 0 – Indicates that the value contained in a data item is not null.
- 1 – If the field is nullable, indicates that the data item contains a null value.

**Note**: Teradata Database returns an error message if imported data has a null value in a field that is not nullable.

The DataInfo parcel, which immediately precedes the first response row returned by Teradata Database, contains information on the total number of columns returned by a request, and the data type and length of each column.
Specifying .EXPORT INDICDATA automatically sets Indicator Mode. Specifying .EXPORT REPORT, EXPORT RESET, .SET RECORDMODE OFF, or .SET INDICDATA OFF automatically resets the default Field Mode.

If the INDICDATA command has not been used, the value is OFF by default. If the command is used and ON or OFF is not specified, BTEQ assumes ON.

You can use the INDICDATA command in a Teradata SQL macro.

**Example 1**

To return data to a client system in native format, type:

```
.database personnel;
.set defaults
.set indicdata on
select empno,
    proj_id from charges
where empno = 10004
order by proj_id
;
.indicdata off
= 1
.defaults
.format off
.logoff
.exit
```

**BTEQ Response**

```plaintext
*** New default database accepted .
*** Success, Stmt# 1 ActivityCount = 2
*** Query completed. 2 rows found. 2 columns returned.
*** Dump of Parcel DATAINFO: 0000 0002 01F4 0002
01C4 0008 * ...4...D..*
*** Record#1 . Dump Of Data: 0000 0027 14C5 D5C7
60F0 F0F0 F2 * ...ENG-0002*
*** Record#2 . Dump Of Data: 0000 0027 14C5 D5C7
60F0 F0F0 F3 * ...ENG-0003*
*** Query completed. 2 rows found. 2 columns returned.
Employee Project
Id Id
------- -------
10004 ENG-0002
10004 ENG-0003
```

**Example 2**

The Example 1 command in a Teradata SQL macro appears as:

```
ECHO '.SET INDICDATA ON';
```
Example 3

To return data in Field Mode after the INDICDATA command option has been set to ON, type:

```
.SET INDICDATA OFF
```
**Purpose**
Identifies the point at which BTEQ resumes processing, as specified in a previous GOTO command.

**Syntax**
```
.LABEL labelname
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>labelname</code></td>
<td>the name given to the statement.</td>
</tr>
<tr>
<td></td>
<td>For network-attached systems, <code>labelname</code> must begin with a letter (either uppercase or lowercase). For channel-attached systems, <code>labelname</code> must begin with a letter or number. Remaining characters can be any combination of letters or numbers.</td>
</tr>
<tr>
<td></td>
<td><code>labelname</code> cannot include blanks or special characters. Only the first 30 characters are significant.</td>
</tr>
</tbody>
</table>

**Usage Notes**
The LABEL command identifies the target of a GOTO statement.

You can use the LABEL command in a Teradata SQL macro.

**Example 1**
The following command example specifies OCCUPATION as the name of the statement at which BTEQ resumes processing as specified in a previous GOTO command:
```
.LABEL OCCUPATION
```
LARGEDATAMODE

**Purpose**

Enables the use of the Teradata Database Multipart Indicator response mode for inline mode retrieval of Large Object (LOB) data. BTEQ limits the record size for exported files to approximately 64K (65473 for workstation builds and 64260 for mainframe builds).

If more than 64K is required, SET LARGEDATAMODE allows hex-dump style output (similar to Record Mode) directed to standard output.

**Syntax**

```
SET LARGEDATAMODE [ON | OFF]
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFF</td>
<td>OFF is the initial setting.</td>
</tr>
<tr>
<td>ON</td>
<td>If you use .SET LARGEDATAMODE without specifying ON or OFF, ON is assumed. When the setting is ON, the only valid data format to specify for the EXPORT command is INDICDATA. EXPORT RESET returns BTEQ to Field Mode for responses.</td>
</tr>
</tbody>
</table>

**Example**

In this example, the sequence of statements trigger Multipart Indicator Mode for an export of data from workstation BTEQ.

```
SET LARGEDATAMODE ON
.EXPORT INDICDATA FILE=MyResults
SELECT MySmallCLOB FROM MyTable;
```

If an attempt is made to retrieve a row of data that exceeds 65473 bytes for workstation builds or 64260 bytes for mainframe builds, the request is aborted with an error similar to the following, where XXX is the row that was too large and YYY is the maximum output record size for the BTEQ build:

```
*** Error: Row data for row# XXX exceeds maximum output record size of YYY.
```
Chapter 5: BTEQ Commands

LARGEDATAMODE

Example

When the LARGEDATAMODE setting is ON, and data is directed to standard out instead of an export file, BTEQ does not impose size limits. The resulting hex-dump output is similar to the output produced for the RECORDMODE command. The main difference is that one or more MultipartRecord parcels may be dumped for each row instead of one Record parcel.

When more than one MultipartRecord parcel is returned, the data dump for the row is continued. For example:

```plaintext
*** Record#1. Dump Of Data:
0000  0030 3510 2700 0000  0000 00AA BBCC 0000 *.05.'............*
0010  0000 0000 0000 0000  0000 0000 0000 0000 *................*
2710 *0000 0000 0000 0000 DDEE FF '*............*
*** Record#1. Dump Of Data Continued:
0000  1027 0000 1027 0000  0000 0000 AABB CC11 +.'...'............*
0010  1111 1111 1111 1111  1111 1111 1111 1111 +................*
2710 +1111 1111 1111 1111  11DD EEFF '*............*
*** Record#1. Dump Of Data Continued:
0000  1027 0000 1027 0000  0000 0000 AABB CC22 +.'...'............*
0010  2222 2222 2222 2222  2222 2222 2222 2222 +................*
2710 *2222 2222 2222 2222  22DD EEFF +''''''''''''''''''''
*** Record#1. Dump Of Data Continued:
0000  1027 0000 +.'..*
```
LOGMECH and LOGDATA

Purpose

The LOGMECH and LOGDATA commands enable specification of the desired security mechanism and any other parameters needed by that mechanism (beyond userid and password) to authenticate the user associated with subsequent LOGON command use.

Valid values for LOGMECH are a single mechanism name up to 8 characters in length. For LOGDATA, valid values are a single mechanism data values up to 32000 bytes in length. The initial value for LOGMECH is 8 spaces and indicates that the default mechanism should be used. When either command is used without specifying a value, its associated initial value is restored.

Syntax

```
.LOGDATA  logdata_string  ;
```

where

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>logdata_string</td>
<td>indicates the parameters for the logon mechanism specified using the LOGMECH command. For information about the logon parameters for supported mechanisms, see Security Administration.</td>
</tr>
</tbody>
</table>

Syntax

```
.LOGMECH  logmech_name  ;
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>logmech_name</td>
<td>defines the logon mechanism. For a discussion of supported logon mechanisms, see Security Administration. The name is limited to 8 characters; it is not case sensitive.</td>
</tr>
</tbody>
</table>
**Usage Notes**

Because the LOGDATA argument is considered sensitive information, BTEQ (in interactive mode) prompts for a value, which is specified in protected mode (can’t see what you are typing). The value cannot be supplied as an argument to the LOGDATA command.

The results for SHOW CONTROLS command used does not reflect the existence of the LOGDATA setting.

For more information about using security mechanisms, see *Security Administration*.

**Examples**

If used, the LOGDATA and LOGMECH commands must precede the LOGON command. The commands themselves may occur in any order. The example is for non-interactive mode use.

The example demonstrates using the LOGDATA, LOGMECH, and LOGON commands in combination to specify the Windows logon authentication method and associated parameters:

```
.logmech NTLM;
.logdata joe@domain1@mypassword;
.logon mydbs;
```
LOGOFF

**Purpose**
Terminates your Teradata Database sessions without exiting from BTEQ.

**Syntax**

```
.LOGOFF
```

**Usage Notes**
Using the LOGOFF command does not exit BTEQ. Use the EXIT or QUIT commands, which are synonymous within BTEQ, if you want to exit BTEQ after logging off Teradata Database session.

(For your own convenience in writing scripts, you may prefer to use the EXIT command after an explicit LOGOFF command, and use the QUIT command when the LOGOFF command is not explicitly provided.)

**Note:** The LOGOFF command does not reset the FORMAT command default values. If you log off from the initial BTEQ session and then specify another LOGON command, the resulting sessions inherit the format values from the prior session.

You cannot use the LOGOFF command in a Teradata SQL macro.

**Example**
To log off of Teradata Database, type:

```
.logoff
```
LOGON

Purpose
Connects you to a Teradata Database.

Syntax
For Interactive Mode:

```
LOGON tdpid/ userid acctid
```

For Batch Mode:

```
LOGON userid tdpid/ password acctid
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>acctid</td>
<td>the account identifier associated with the userid</td>
</tr>
<tr>
<td>password</td>
<td>the password associated with the userid.</td>
</tr>
<tr>
<td>tdpid/</td>
<td>the identifier associated with a particular Teradata server</td>
</tr>
<tr>
<td>userid</td>
<td>a user identifier</td>
</tr>
</tbody>
</table>

User Logon Exit Routine
You can use the User Logon Exit routine of the Call-Level Interface to make some or all elements of the logon string optional.

Using LOGON
When using BTEQ in batch mode or with input files, enter the entire LOGON command on one line. When using BTEQ interactively, you cannot enter a password or account id on the same line as the userid.

If you use a LOGON command when you are already logged on, BTEQ logs you off the first session and then logs you on for a new session using the userid, tdpid, and acctid specifications of the new LOGON command. Note, however, that the new session inherits the control settings, such as page length and headings, from the old session.
When you execute the LOGON command, the number of Teradata sessions previously determined by the SESSIONS command takes effect. You must use the SESSIONS and TDP commands before using the LOGON command for them to take effect. If you specify more than one session and one logon fails, BTEQ continues with the successful sessions.

You cannot use the LOGON command in a Teradata SQL macro.

The Teradata Database imposes restrictions on the size of object names used within LOGON requests. Refer to the Teradata Database documentation for details on the actual limits and the ramifications for object names that contain multi-byte characters.

When you need to specify userid and/or password values that contain the characters used to delimit the individual components of the LOGON string, such as a comma, the values must be surrounded by matched quotes.

Using the Optional tdpid Identifier

Use the optional tdpid identifier to specify a particular Teradata Database. See your system or site administrator for the identifier associated with the Teradata Database that you plan to use. If you do not specify a tdpid and the site administrator has not updated the System Parameter Block, the default identifier is DBC for network-attached clients.

When you specify a tdpid in a LOGON command, that tdpid becomes the default tdpid for any future LOGON commands used within the same BTEQ executioner. To change the default tdpid, use a TDP command, or specify a new tdpid in another LOGON command.

The tdpid identifier specification, in effect, is optional if your installation has only one TDP, if you have previously executed a TDP command, or if you choose the default TDP. The value for the tdpid specification is not case-sensitive; you can use either uppercase or lowercase characters.

If you do not provide the tdpid, BTEQ assumes you are using a predefined default tdpid, (if this is the first logon). If you are already logged on, BTEQ uses the current tdpid as the default and prompts you for the userid. BTEQ accepts the tdpid if you do not provide the userid with the LOGON command.

Note: An IPv4 or IPv6 address can be entered as part of the tdpid. For details, see “Entering IPv4 and IPv6 Network Addresses” on page 52.

The User ID and Password Prompts

When using BTEQ interactively, you can log on by entering only a tdpid, which may be followed by a semicolon or a blank space. If BTEQ accepts the tdpid, it clears the input area and prompts for a userid:

Userid:

The userid can be followed by either a semicolon, a comma, or a blank space. If BTEQ accepts the logon string, it clears the input area and prompts for a password:

Password:

Supply the password associated with the userid, and specify the acctid on the same line following the password, if it is required. The password cannot include or end with a semicolon
unless the entire value is enclosed in quotes. The acctid may be followed by a semicolon or a blank space.

Note: On channel-attached systems, you can omit your password if your site has an exit routine to add the password. On network-attached systems, you cannot omit your password from the logon string.

Using the Optional acctid Identifier
The acctid string can contain special characters, but they might be interpreted differently by different output devices. A script containing special characters might have to be modified if you route your output to another device. Therefore, do not use special characters in the acctid string.

If the acctid has an apostrophe (single quote) character, then either use the second form of the LOGON command (the one with quotes as delimiters) or double the apostrophe character, as follows:

```
.LOGON 0/fml,fml, "engineering's account"
```

or

```
.LOGON 0/fml,fml,'engineering''s account'
```

If the acctid does not have an apostrophe, the two forms of the LOGON command are the same.

Example 1
To specify the account value as part of an interactive LOGON statement:

```
.LOGON tdpid/user,,'acctid';
```

(Essentially, the two consecutive commas instruct BTEQ to prompt for the password.)

Example 2
To supply the account value as part of the password entry in an interactive LOGON, use the syntax:

```
passwd, 'accountid'
```

If You Enter an Incorrect Parameter
If you enter any parameter incorrectly, the logon fails and BTEQ returns an error message. For security reasons, the error message does not state in which parameter the error occurred.

Effects of the LOGONPROMPT Command
The LOGONPROMPT command allows you to bypass the prompts and warnings related to conventional LOGON command use. If the LOGONPROMPT setting is ON, normal LOGON command behavior results.

If the LOGONPROMPT setting is OFF:

- BTEQ does not prompt for userid or password if you omit these entries during logon. Therefore, if you want to enter a conventional logon string, you must fully qualify the LOGON command.
You can specify the following before establishing a session:
- `=` command
- `COMPILE` command
- `REPEAT` command
- SQL statement

**Example**

Below is an example of setting `LOGONPROMPT` to `OFF`, then logging on with a `tdpid` value of `slugger`.

```
.SET LOGONPROMPT OFF;
.LOGON slugger/
```

**BTEQ Response**

```
*** Logon successfully completed.
```

**Security Concerns**

If you are concerned about the security of your password on a channel-attached system, you can alter your JCL to accept the `LOGON` command from another data set/file under the control of ACF2 or another client-resident security system. For example:

```
//stepname EXEC PGM=BTQMAIN,...
//SYSIN DD DSN=... (data set containing .LOGON command)
//
```

You can then log on by simply entering the `LOGON` command with a valid `userid` and no password if your System Administrator has granted this option.

**Using Connect_Wait to Decrease Wait Time**

When a host group has one or more disabled node, AP, or COP, the time it takes to log on for BTEQ sessions increases and can become excessive when CLI attempts to connect multiple sessions through the disabled processors. Use the `clispb.dat` option, `connect_wait`, to decrease the amount of time that CLIv2 waits for responses before polling another node, AP, or COP.

See Teradata Call-Level Interface Version 2 Reference for Channel-Attached Systems or Teradata Call-Level Interface Version 2 Reference for Network-Attached Systems for information about using the User Logon Exit routine or the `connect_wait` option.

**Example 1**

To log on to BTEQ in batch mode as user ABC with `ABC` as the password (which is masked from view on the output listing), specify the `LOGON` command on one line:

```
.logon ABC,ABC
```

**BTEQ Response**

After you specify the `LOGON` command, BTEQ displays:

```
*** Logon successfully completed.
*** Total elapsed time was 3 seconds.
```
**Example 2**
To log on to BTEQ interactively as user ABC, specify the following LOGON command:

```
.logon ABC
```

**BTEQ Response**
After you specify the first part of the LOGON command, BTEQ prompts for the password:

```
Password:
```
After supplying your password, which is not displayed, BTEQ displays:

```
*** Logon successfully completed.
*** Total elapsed time was 3 seconds.
```
LOGONPROMPT

Purpose
Allows you to bypass the prompts and warnings related to conventional LOGON command use. The LOGONPROMPT command enables:

- Optional use of the LOGON command before submitting SQL
- Suppression of the prompt to qualify userid and password information for the LOGON command

Note: You can only use the LOGONPROMPT command when running BTEQ on network-attached systems.

Syntax

```
. SET LOGONPROMPT [ON | OFF]
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFF</td>
<td>BTEQ does not prompt for logon string values when you specify a LOGON command without specifying a user name. If no option is specified, OFF is taken by default.</td>
</tr>
<tr>
<td>ON</td>
<td>BTEQ prompts you to use the LOGON command and specify valid logon string values to establish sessions. If you supply an SQL statement before establishing a session, BTEQ warns you to log on first, then discards the SQL statement. This is the initial value when you first start the program.</td>
</tr>
</tbody>
</table>

Usage Notes
If you set LOGONPROMPT to OFF, BTEQ does not prompt you for userid or password if you omit these entries during logon. Therefore, if you want to supply a conventional logon string, you must fully qualify the LOGON command.

When the LOGONPROMPT setting is OFF, you can specify the following before establishing a session:

- SQL statement
- = command
Chapter 5: BTEQ Commands

LOGONPROMPT

- REPEAT command
- COMPILE

After you specify the command or statement, BTEQ attempts to establish the session by using the default *tdpid* and assumed logon string information before it submits the command or statement.
MAXERROR

**Purpose**
Designates a maximum error severity level beyond which BTEQ terminates job processing.

**Syntax**
```
SET MAXERROR n
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>the error severity level at which BTEQ terminates.</td>
</tr>
</tbody>
</table>

**Usage Notes**
If a BTEQ script contains a Teradata SQL statement that produces an error severity level greater than the designated MAXERROR value, BTEQ immediately aborts. Therefore, if you want BTEQ to exit on an error severity level of 8, set the MaxErrorLevel to 7.

If you do not specify a MAXERROR value, the automatic termination feature is disabled and the BTEQ job continues to execute until BTEQ encounters one of the following conditions:

- End-of-file for the primary command input file
- A QUIT command
- A fatal error
  - When BTEQ receives an I/O abend, system error messages appear in the z/OS JES job log.
  - When BTEQ receives an I/O error or an abend, the SAS/C runtime library produces an LSCX message that may provide more information about the error.

**Note:** For more information on I/O errors and abends, refer to “I/O Errors and Abends” on page 92. For more information about error severity levels, see “ERRORLEVEL” on page 147.
MESSAGEOUT

Purpose
Specifies the name of a file for BTEQ to store the messages that would normally be sent to the standard output.

Syntax
For Channel-Attached Systems:

\[ \text{.MESSAGEOUT} \text{ FILE} = \text{xxx} \]

For Network-Attached Systems:

\[ \text{.MESSAGEOUT} \text{ BOM|NOBOM} \text{ FILE} = \text{xxx} \]

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>xxx</td>
<td>the name of the z/OS JCL DD or UNIX file name of the file to receive data that would normally be sent to the standard output</td>
</tr>
<tr>
<td>BOM</td>
<td>NOBOM</td>
</tr>
</tbody>
</table>

Usage Notes
If you specify an existing file, BTEQ appends data to that file. Otherwise, a new file is created. You may use any device name that is valid on your system as a file name.

BTEQ closes the MESSAGEOUT file when one of the following is used:

- a MESSAGEOUT command specifying another message file
- an EXIT/QUIT command
BTEQ does not close the file until the last file record has been written. If you read the file before it is closed, the last few records may be missing. They are added when the file is closed in response to your MESSAGEOUT or EXIT/QUIT command.

For network-attached systems, when BTEQ is started with the -m command line option, stdio will be based on the system locale. Therefore, the MESSAGEOUT file will contain locale-specific characters (instead of UTF8/UTF16 encoded characters) and will not contain a BOM.

**Example 1**

For z/OS, to redirect messages to the DDNAME MYOUT use the following command:

```
.MESSAGEOUT DDNAME=MYOUT
```

where DDNAME was previously defined with a TSO command. For example,

```
.TSO ALLOCATE DDNAME(MYOUT) DSNAME(YOUR.MSG.OUT) SHR
```

**Example 2**

From a network-attached system, to redirect standard output to the file OUT1, use the following command:

```
.MESSAGEOUT FILE=OUT1
```
**NOTIFY**

**Purpose**

Performs a user exit or pre-defined action when certain significant events occur.

The NOTIFY command is provided for use in an operator-free environment, where job scheduling relies heavily on automation to optimize system performance. The function provides hooks that system programmers can use to automate BTEQ job streams fully so they can be integrated with third party vendor products, removing the need to parse the output of BTEQ scripts.

For example, by writing an exit in C (without using CLIv2) and using the NOTIFY...EXIT option, a programmer can provide a means for an automation package to detect that a BTEQ request either succeeds or fails, how many blocks were returned in a successful request, what the return code was for a failed request, and so on.

NOTIFY only applies to the request which immediately follows it.

**Syntax**

```
SET NOTIFY
OFF       EXIT name
LOW       MSG text
MEDIUM    QUEUE options
HIGH

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXIT</td>
<td>A user-written exit is to be called at the appropriate time</td>
</tr>
<tr>
<td>HIGH</td>
<td>Notification is to be provided at all events detailed in</td>
</tr>
<tr>
<td>LOW</td>
<td>Notification is to be provided for one event only.</td>
</tr>
<tr>
<td></td>
<td>This option:</td>
</tr>
<tr>
<td></td>
<td>• is provided for use with the EXIT, MSG, and QUEUE options</td>
</tr>
<tr>
<td></td>
<td>• cannot be used in a .repeat loop</td>
</tr>
<tr>
<td>MEDIUM</td>
<td>Notification is to be provided at points of typical user interest, such as:</td>
</tr>
<tr>
<td></td>
<td>• when the job finishes</td>
</tr>
<tr>
<td></td>
<td>• when the most significant event occurs</td>
</tr>
</tbody>
</table>
```
Chapter 5: BTEQ Commands

NOTIFY

Using NOTIFY

You cannot use NOTIFY with the = command.

If you use NOTIFY on a statement which is REPEATed, the notify action is only invoked for the first request, all following requests are ignored.

NOTIFY is not affected by SET DEFAULT.

The NOTIFY option is displayed by SHOW CONTROLS.

You cannot use NOTIFY in a Teradata SQL macro.

The data type for the ActivityCount returned by BTEQ has been changed from Int32 (signed 32-bit integer) to UInt32 (unsigned 32-bit integer). If this variable is used in any way as part of a notify exit routine, change the print specifier for this variable to %u and recompile the routine.
The following table shows Notify Events and their unique numbers:

<table>
<thead>
<tr>
<th>Notify Event</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialization</td>
<td>0</td>
</tr>
<tr>
<td>Request Start</td>
<td>42</td>
</tr>
<tr>
<td>Request Done</td>
<td>43</td>
</tr>
<tr>
<td>Fetch start</td>
<td>44</td>
</tr>
<tr>
<td>Complete</td>
<td>45</td>
</tr>
<tr>
<td>DBS Restart</td>
<td>9</td>
</tr>
<tr>
<td>CLI Error</td>
<td>10</td>
</tr>
<tr>
<td>DBS Error</td>
<td>11</td>
</tr>
<tr>
<td>Exit</td>
<td>12</td>
</tr>
</tbody>
</table>

**Note:** When you create Notify User Exits, you must ensure that they ignore invalid event codes, and that the receipt of those event codes does not cause BTEQ to abend. This allows for the Notify command to be expanded with additional events without affecting any Notify User Exits.

Table 8: Events That Create Notifications

<table>
<thead>
<tr>
<th>Event</th>
<th>Notification Level</th>
<th>Parameter</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>Medium</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Notify processed</td>
<td>No</td>
<td>No</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Request issued</td>
<td>No</td>
<td>No</td>
<td>Request string</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>First request completes successfully</td>
<td>Yes</td>
<td>Yes</td>
<td>Request number, Statement number, Activity count</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ready to fetch answer set</td>
<td>No</td>
<td>No</td>
<td>Request number, Statement number, Activity count</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Done fetching all answer sets/ Request successful</td>
<td>No</td>
<td>No</td>
<td>Total number of requests</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- BTEQ exits</td>
<td>No</td>
<td>Yes</td>
<td>Return code</td>
</tr>
<tr>
<td>- NOTIFY goes out of scope</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBS Restart</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>CLI Error</td>
<td>Yes</td>
<td>Yes</td>
<td>CLI error code</td>
</tr>
</tbody>
</table>
When NOTIFY is processed, it performs an ENQ upon a QUEUE with RNAME of 'TRDUSER' and a scope of 'SYSTEMS'. This call blocks until it acquires the QUEUE.

After the job gets the QUEUE, it continues until it reaches a specific point (such as the request completes) when it releases the QUEUE by performing a DEQ.

**Error Handling Notes**

When an error occurs:

- When NOTIFY is processed, the subsystems used by BTEQ are initialized, and if necessary, any user exits are loaded and a call is made to initialize the system log (or an ENQ is performed).
  
  If initialization fails, a warning message is issued and processing continues. The exception to this rule is the case where a NOTIFY... QUEUE is issued. If initialization (the ENQ call) fails, it is a fatal error.

- If anything fails after initialization, the request fails. If a user exit returns anything other than 0, a failure is indicated and the job stops.

**Restart Notes**

If an application ends abnormally or unsuccessfully, it can be restarted and some NOTIFY-related activities are re-executed. This is an important issue with respect to writing user exits.

If an application ends abnormally or unsuccessfully while it is holding a queue (using the QUEUE type parameter in z/OS), the application releases the queue before exiting the job. Therefore, when the job restarts, you must ensure that the application acquires the queue again before processing continues.

**Building an Exit Module**

When building an exit module, there are general procedures to consider that are constant across all operating systems:

- The exit must be named _dynamn
- Success is indicated by the return of a 0 (32-bit signed integer format)
- Failure is indicated by the return of a nonzero value (32-bit signed integer format). Use different integers to indicate different errors.
- The only parameter to the procedure is a pointer to a variable record structure
- A C prototype example for an exit procedure might be as follows:

  ```c
  Int32 _dynamn(BTNotifyExitParm *P)
  ```

- The C structure that is passed to user exits by BTEQ for NOTIFY can be obtained from the sample notify header file btnfyext.h which is included in the BTEQ package

---

**Table 8: Events That Create Notifications (continued)**

<table>
<thead>
<tr>
<th>Event</th>
<th>Notification Level</th>
<th>Parameter</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBS Error</td>
<td>Yes</td>
<td>Yes</td>
<td>DBS error code</td>
</tr>
</tbody>
</table>
Table 9 summarizes how to build exit modules using different operating systems:

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sun Solaris</td>
<td>Compile the module with the following options: <code>-K PIC</code> <code>-G</code>.</td>
</tr>
<tr>
<td></td>
<td>The resulting module is in shared library format.</td>
</tr>
<tr>
<td></td>
<td>The exit must be specified using a filesystem path descriptor unless the environment variable <code>LD_LIBRARY_PATH</code> is set to point to the module at run time.</td>
</tr>
<tr>
<td></td>
<td>BTEQ installs three sample files: the sample C file, include file, and makefile. The respective files are located in:</td>
</tr>
<tr>
<td></td>
<td>* /opt/teradata/client/sample/btnfyext.c</td>
</tr>
<tr>
<td></td>
<td>* /opt/teradata/client/include/btnfyext.h</td>
</tr>
<tr>
<td></td>
<td>* /opt/teradata/client/sample/mkbtnfyext.unx</td>
</tr>
<tr>
<td></td>
<td>To run the sample makefile, specify:</td>
</tr>
<tr>
<td></td>
<td><code>make -f mkbtnfyext.unx</code></td>
</tr>
<tr>
<td>Red Hat Enterprise Linux</td>
<td>Compile the module with the following options: <code>-m32</code> <code>-fPIC</code> <code>-shared</code>.</td>
</tr>
<tr>
<td></td>
<td>The resulting module is in shared library format.</td>
</tr>
<tr>
<td></td>
<td>The exit must be specified using a filesystem path descriptor unless the environment variable <code>LD_LIBRARY_PATH</code> is set to point to the module at run time.</td>
</tr>
<tr>
<td></td>
<td>BTEQ installs three sample files: the sample C file, include file, and makefile. The respective files are located in:</td>
</tr>
<tr>
<td></td>
<td>* /opt/teradata/client/sample/btnfyext.c</td>
</tr>
<tr>
<td></td>
<td>* /opt/teradata/client/include/btnfyext.h</td>
</tr>
<tr>
<td></td>
<td>* /opt/teradata/client/sample/mkbtnfyext.linux</td>
</tr>
<tr>
<td></td>
<td>To run the sample makefile, specify:</td>
</tr>
<tr>
<td></td>
<td><code>make -f mkbtnfyext.linux</code></td>
</tr>
<tr>
<td>z/Linux</td>
<td>Compile the module using the following options:</td>
</tr>
<tr>
<td></td>
<td>* <code>-m31</code></td>
</tr>
<tr>
<td></td>
<td>* <code>-fPIC</code></td>
</tr>
<tr>
<td></td>
<td>* <code>-shared</code></td>
</tr>
<tr>
<td></td>
<td>The resulting module is in shared library format.</td>
</tr>
<tr>
<td></td>
<td>The exit must be specified using a filesystem path descriptor unless the environment variable <code>LD_LIBRARY_PATH</code> is set to point to the module at runtime.</td>
</tr>
<tr>
<td></td>
<td>BTEQ installs three sample files, which can be found in the following locations:</td>
</tr>
<tr>
<td></td>
<td>* C source file</td>
</tr>
<tr>
<td></td>
<td><code>/opt/teradata/client/sample/btnfyext.c</code></td>
</tr>
<tr>
<td></td>
<td>* Header (include) file</td>
</tr>
<tr>
<td></td>
<td><code>/opt/teradata/client/include/btnfyext.h</code></td>
</tr>
<tr>
<td></td>
<td>* Makefile</td>
</tr>
<tr>
<td></td>
<td><code>/opt/teradata/client/sample/mkbtnfyext.suselinux-390.32</code></td>
</tr>
<tr>
<td></td>
<td>To run the sample makefile:</td>
</tr>
<tr>
<td></td>
<td><code>make -f mkbtnfyext.suselinux-390.32</code></td>
</tr>
</tbody>
</table>
Table 9: Building Exit Modules Using Different Operating Systems  (continued)

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Procedure</th>
</tr>
</thead>
</table>
| IBM AIX          | Compile the module with the following option: -G  
The resulting module is in shared library format.  
The exit must be specified using a file system path descriptor unless the environment variable LD_LIBRARY_PATH is set to point to the module at run time.  
BTEQ installs three sample files: the sample C file, include file, and makefile. The respective files are located in:  
* /opt/teradata/client/sample/btnfyext.c  
* /opt/teradata/client/include/btnfyext.h  
* /opt/teradata/client/sample/mkbtnfyext.aix  
To run the sample makefile, specify:  
make -f mkbtnfyext.aix |
| Hewlett-Packard HP-UX PA-RISC | Compile the module with the following options: +z +u1  
The resulting module is in shared library format.  
The exit must be specified using a file system path descriptor unless the environment variable SHLIB_PATH is set to point to the module at run time.  
BTEQ installs three sample files: the sample C file, include file, and makefile. The respective files are located in:  
* /opt/teradata/client/sample/btnfyext.c  
* /opt/teradata/client/include/btnfyext.h  
* /opt/teradata/client/sample/mkbtnfyext.hpux  
To run the sample makefile, specify:  
make -f mkbtnfyext.hpux |
| (ia64) Hewlett-Packard HP-UX | Compile the module with the following options: +u1 -D_REENTRANT +DD64  
The resulting module is in shared library format.  
The exit must be specified using a file system path descriptor unless the environment variable LD_LIBRARY_PATH/SHLIB_PATH is set to point to the module at run time.  
BTEQ installs three sample files: the sample C file, include file, and makefile. The respective files are located in:  
* /opt/teradata/client/sample/btnfyext.c  
* /opt/teradata/client/include/btnfyext.h  
* /opt/teradata/client/sample/mkbtnfyext.hpux-ia64  
To run the sample makefile, specify:  
make -f mkbtnfyext.hpux-ia64 |
Example 1 - UNIX

This example uses a user-loadable module called btnfyext.so. This module is built from the file btnfyext., which is included along with a makefile in the BTEQ distribution as an example.

Before calling BTEQ, set the environmental variable LD_LIBRARY_PATH so that the module can be found:

```
Script started on Wed Jan 24 18:19:12 1996
app002-0:/home/cme/src/notify/clientsw/src/bteqsrc > export
LD_LIBRARY_PATH=/home/cme/src/notify/clientsw/src/bteqsrc
app002-0:/home/cme/src/notify/clientsw/src/bteqsrc > bteq
Teradata BTEQ 3.1 for UNIX5. Enter your logon or BTEQ command:
.logon dbc
....
```

```
BTEQ -- Enter your SQL request logon or BTEQ command:
.set notify low exit btnfyext.so
.set notify low exit btnfyext.so
BTEQ -- Enter your SQL request logon or BTEQ command:
sel * from sessiontbl;
sel * from sessiontbl;
*** Query completed. 5 rows found. 24 columns returned.
*** Total elapsed time was 1 second.
HostNo SessionNo IFPNo LogonDate LogonTime LogonCollation LogonData
------- ---------- ----- --------- ----------- -------------- ----
 97         -2   4-3  96/01/24 15:25:29.56 H    TDPUSER
 93         -2   3-4  96/01/09 17:13:25.24 H    TDPUSER
100        1,012 4-4  96/01/24 18:22:18.20 H    DBC
 93         -3   3-4  96/01/09 17:13:25.25 H    TDPUSER
 97         -3   3-3  96/01/24 15:25:29.57 H    TDPUSER
....
```
The user-loadable module writes the parameters it passed to a file called `NFYEXT.OUT`. After the parameter sequence, the file contains one line:

```
exit called @ bteq request complete.
```

**Example 2 - UNIX**

This example uses the MEDIUM option. The BTEQ output is not included because it is similar to the output from Example 1.

The `NFYEXT.OUT` output file contains two lines:

```
exit called @ bteq request complete.
exit called @ bteq notify out of scope: return code 0.
```

**Example 3 - UNIX**

This example uses the HIGH option. Every event is used, and the exit is called six times:

```
exit called @ bteq init.
exit called @ bteq start request: 'sel * from sessiontbl;'.
exit called @ bteq request complete.
exit called @ bteq fetch: statement 1, request 1 activity 5.
exit called @ bteq request processing complete: 1 requests
exit called @ bteq notify out of scope: return code 0.
```

**Example 4 - UNIX**

This example uses the MSG option.

The STDOUT from BTEQ is as follows:

```
BTEQ 3.1 Wed Jan 24 20:05:10 1996
+---------+---------+---------+---------+---------+-----
  .logon dbc,  
  *** Logon successfully completed.  
  *** Transaction Semantics are BTET.  
  *** Character Set Name is ASCII.  
  *** Total elapsed time was 1 second.  
+---------+---------+---------+---------+---------+-----
  .set notify high msg 'This is a UNIX msg';
  +---------+---------+---------+---------+---------+-----
  sel * from sessiontbl;
  *** Query completed. 5 rows found. 24 columns returned.  
  *** Total elapsed time was 1 second.  
  HostNo SessionNo IFPNo LogonDate   LogonTime LogonCollation LogonData  
  ------- --------- ----- --------- ----------- -------------- ---------  
   97 -2   4-3  96/01/24 15:25:29.56 H TDFUSER  
   93 -2   3-4  96/01/09 17:13:25.24 H TDFUSER  
   93 -3   3-4  96/01/09 17:13:25.25 H TDFUSER  
  100 1,013   4-4  96/01/24 20:02:59.95 H DBC  
   97 -3   3-3  96/01/24 15:25:29.57 H TDFUSER  
+-----+--------_-+------+---------+-----------+-----------------------
  .quit  
  *** You are now logged off from the DBC.  
  *** Exiting BTEQ...  
  *** RC (return code) = 0
```
For UNIX, the MSG option asks BTEQ to write a message using the standard syslog system (the message is written to a standard system log).

On the test system, the default location for the file is `/var/adm/usererr/error.01-24`. However, the location of this file can and does vary.

The above job produces the following lines in the system log:

Jan 24 20:05:10 app002-0 unix: syslog: This is a UNIX msg - BTEQ notify processed.
Jan 24 20:05:10 app002-0 unix: syslog: This is a UNIX msg - BTEQ request issued: 'sel * from sessiontbl;'.
Jan 24 20:05:11 app002-0 unix: syslog: This is a UNIX msg - BTEQ request complete.
Jan 24 20:05:11 app002-0 unix: syslog: This is a UNIX msg - BTEQ ready to fetch request 1, statement 1: activity count 5.
Jan 24 20:05:11 app002-0 unix: syslog: This is a UNIX msg - BTEQ fetches complete: 1 requests.
Jan 24 20:05:11 app002-0 unix: syslog: This is a UNIX msg - BTEQ notify out of scope RC = 0.

Example 5 - z/OS
This example uses the ENQUEUE/DEQUEUE function, which is only supported by z/OS.

The SYSPRINT is as follows:

```
+---------+---------+---------+---------+---------+----
//**************************************|
/* just a little test program...      */
//**************************************|
.WIDTH 80
+---------+---------+---------+---------+---------+----
.LOGON tdri/cme,
 *** Logon successfully completed.

*** Total elapsed time was 0.75 seconds.
```

```
+--------------------------------------------+
*.show versions;
BTEQ Version 5.3.0 for IBM OS/370
BTQMain   : H3_03
BTQUtil   : H3_05
BTQResp   : H3_03
BTQParse  : H3_02
BTQNotfy  : H3_00
CapAAUtl  : H3_01
CapCLUtl   : H3_02
CapI0Utl   : H3_03
CapLogW   : H3_00
CapMVSQ   : H3_00
CapNfy    : H3_00
CapLoadM  : H3_00
MOSIDEP   : H0_02
CapTypes  : H2_01
CapDBC    : H3_00
+--------------------------------------------+
```
ITBDBC : H3_02
MOSIIF : H2_01
BTEQ linking date is 01/03/96

.set notify low queue rname 'sighsighsighsigh';

.if errorcode <> 0 then .quit errorcode;

sel * from cme.foo;
*** Query completed. 4 rows found. One column returned.
*** Total elapsed time was 0.27 seconds.
96/01/18 sel * from cme.foo; Page
1

LOGOFF
*** You are now logged off from the DBC.
*** Exiting BTEQ...
Ending BTEQ Thu Jan 18 15:03:42 1996

The JES2 LOG output follows. Note the second, third and fourth lines. The three messages are written when a queue is used:

- TRDF0001I, indicates that the job is waiting for a queue whose QNAME is 'TRDFEQUE' and whose RNAME is 'sighsighsighsigh'.
- TRDF0002I, indicates that the queue is now held by the job. It is possible that the job might block for an indefinite period of time between the TRDF000 and TRDF0002I messages.
- TDRF0003I, indicates that the job is releasing the queue. These messages can help users of the QUEUE feature debug their jobs.

15.03.29 JOB 3207 $HASP373 BTEQTEST STARTED - INIT 42 - CLASS A - SYS MVS2
15.03.41 JOB 3207 +TRDF0001I BTEQTEST ENQ - Q=TRDFEQUE R=sighsighsighsigh
15.03.41 JOB 3207 +TRDF0002I BTEQTEST ENQ - Q=TRDFEQUE R=sighsighsighsigh
15.03.42 JOB 3207 +TRDF0003I BTEQTEST DEQ - Q=TRDFEQUE R=sighsighsighsigh
15.03.42 JOB 3207 BTEQTEST.STEP01 . RC=ZERO T=0000 D=0210 MT=.17S ET=012S
15.03.42 JOB 3207 BTEQTEST ENDED. NAME=CHARLES EUBANKS TOTAL MT=.17S ET=012S
15.03.42 JOB 3207 $HASP395 BTEQTEST ENDED
------ JES2 JOB STATISTICS ------
18 JAN 96 JOB EXECUTION DATE
38 CARDS READ
163 SYSOUT PRINT RECORDS
0 SYSOUT PUNCH RECORDS
9 SYSOUT SPOOL KBYTES
0.21 MINUTES EXECUTION TIME

*** Query completed. 5 rows found. 24 columns returned.
Example 6 - z/OS
This example uses the MSG option.

The SYSPRINT is as follows:

```
TERADATA CORPORATION            BTEQ 5.3.0 Thu Jan 18 15:05:11 1996

+---------+---------+---------+---------+---------+----
//***************************************/
/* just a little test program...     */
/***************************************/
.WIDTH 80
+---------+---------+---------+---------+---------+----
.LOGON tdri/cme,
*** Logon successfully completed.

*** Total elapsed time was 0.65 seconds.

+---------+---------+---------+---------+---------+----
.set notify low msg 'this is a test';
+---------+---------+---------+---------+---------+----
.if errorcode <> 0 then .quit errorcode;
+---------+---------+---------+---------+---------+----
.sel * from cme.foo;
*** Query completed. 4 rows found. One column returned.
*** Total elapsed time was 0.17 seconds.
96/01/18                      sel * from cme.foo;                     Page
1
   a
   ------------
   2
   4
   1
   3
+---------+---------+---------+---------+---------+----
.LOGOFF
*** You are now logged off from the DBC.
+---------+---------+---------+---------+---------+----
*** Exiting BTEQ...
Ending BTEQ  Thu Jan 18 15:05:12 1996

The JES2 LOG output follows. The TRDF0010I message is the MSG output, thus it contains
the user-provided string “this is a test”:

15.05.06 JOB 3214  $HASP373 BTEQTEST STARTED - INIT 42 - CLASS A - SYS
MVS2
15.05.11 JOB 3214  +TRDF0010I this is a test - BTEQ request complete.'
15.05.12 JOB 3214  BTEQTEST.STEP01 .      RC=ZERO T=0000 D=0214
MT=.16S ET=5.5S
15.05.12 JOB 3214  BTEQTEST ENDED. NAME=CHARLES EUBANKS      TOTAL
MT=.16S ET=5.6S
15.05.12 JOB 3214  $HASP395 BTEQTEST ENDED
------- JES2 JOB STATISTICS -------
Example 7 - Windows

This example uses a user-loadable module called *btnfyext.dll*. The module is built from *btnfyext.c*, which is included with a makefile in the BTEQ distribution as an example.

```
c:\teradata\bin>bteq
Teradata BTEQ 04.00.01.00 for WIN32. Enter your logon or BTEQ command:
  .logon weekly
  .logon weekly
Password:
***Logon successfully completed.
***Transaction Semantics are BTET.
***Character Set Name is ASCII.
***Total elapsed time was 32 seconds.
BTEQ -- Enter your SQL request logon or BTEQ command:
  .set notify low exit btnfyext.dll
BTEQ -- Enter your SQL request logon or BTEQ command:
  sel * from dbc.sessioninfo;
***Query completed. 7 rows found. 15 columns returned.
***Total elapsed time was 1 second.
```

<table>
<thead>
<tr>
<th>UserName</th>
<th>AccountName</th>
<th>Session</th>
</tr>
</thead>
<tbody>
<tr>
<td>WEEKLY</td>
<td>DBC</td>
<td>1,2</td>
</tr>
<tr>
<td>WEEKLY</td>
<td>DBC</td>
<td>1,2</td>
</tr>
<tr>
<td>WEEKLY</td>
<td>DBC</td>
<td>1,2</td>
</tr>
<tr>
<td>WEEKLY</td>
<td>DBC</td>
<td>1,0</td>
</tr>
<tr>
<td>WEEKLY</td>
<td>DBC</td>
<td>1,2</td>
</tr>
<tr>
<td>WEEKLY</td>
<td>DBC</td>
<td>1,2</td>
</tr>
<tr>
<td>WEEKLY</td>
<td>DBC</td>
<td>1,2</td>
</tr>
</tbody>
</table>

This user-loadable module writes the parameters it passed to a file called *NFYEXIT.OUT*. After the parameter sequence, the file contains one line:

```
exit called @ bteq init.
```

Example 8 - Windows

This example uses the MEDIUM option. The BTEQ output is not included because it is similar to the output from EXAMPLE 7.

The *NFYEXIT.OUT* output file contains two lines:

```
exit called @ bteq request complete
exit called @ bteq notify out of scope: return code 0
```

Example 9 - Windows

This example uses the HIGH option, and every event that can be used. The exit is called six times:

```
exit called @ bteq init.
exit called @ bteq start request: 'sel * from dbc.sessioninfo;'.
exit called @ bteq request complete.
exit called @ bteq fetch: statement 1, request 1 activity 7.
```
Example 10 - Windows
This example uses the MSG option. The STDOUT from BTEQ is as follows:

c:\teradata\bin>bteq
Teradata BTEQ 04.00.01.00 for WIN32. Enter your logon or BTEQ command:
.logon weekly
_PASSWORD:
***Logon successfully completed.
***Transaction Semantics are BTET.
***Character Set Name is ASCII.
***Total elapsed time was 41seconds.
BTEQ -- Enter your SQL request logon or BTEQ command:
.set notify high msg 'this is a NT msg';
.set notify high msg 'this is a NT msg';
BTEQ -- Enter your SQL request logon or BTEQ command:
sel * from dbc.sessioninfo;
sel * from dbc.sessioninfo;
***Query completed. 7 rows found. 15 columns returned.
***Total elapsed time was 9 seconds.

<table>
<thead>
<tr>
<th>UserName</th>
<th>AccountName</th>
<th>Session</th>
</tr>
</thead>
<tbody>
<tr>
<td>WEEKLY</td>
<td>DBC</td>
<td>1,0</td>
</tr>
<tr>
<td>WEEKLY</td>
<td>DBC</td>
<td>1,2</td>
</tr>
<tr>
<td>WEEKLY</td>
<td>DBC</td>
<td>1,2</td>
</tr>
<tr>
<td>WEEKLY</td>
<td>DBC</td>
<td>1,2</td>
</tr>
<tr>
<td>WEEKLY</td>
<td>DBC</td>
<td>1,2</td>
</tr>
<tr>
<td>WEEKLY</td>
<td>DBC</td>
<td>1,2</td>
</tr>
<tr>
<td>WEEKLY</td>
<td>DBC</td>
<td>1,2</td>
</tr>
</tbody>
</table>

BTEQ -- Enter your SQL request logon or BTEQ command:
.quit;
quit;
***You are now logged off from the DBC.
***Exiting BTEQ...
***RC (return code) = 0

For Windows, the MSG option asks BTEQ to issue an event with a message to EventLog service. This means that the message is trapped and can be viewed by Event Viewer under Application log.
### NULL

**Purpose**
Specifies a character or a character string to represent a null field resulting from an SQL SELECT statement.

**Syntax**

```
SET NULL AS 'string'
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>The value is a single question mark character by default.</td>
</tr>
<tr>
<td></td>
<td>The maximum length of the null string is 253 characters.</td>
</tr>
</tbody>
</table>

**Usage Notes**

The character or string specified by a NULL command during a session remains in effect until a new NULL command is used.

You should not use special characters within the string because special characters may be interpreted differently by different output devices. You might have to modify a script that uses special characters if you route your output to another device.

If the string will have an apostrophe (single quote) character, use the second form of the NULL command (the one with quotes as delimiters) or double the apostrophe. For example:

```
.SET NULL AS "value's missing"
```

or

```
.SET NULL AS 'value''s missing'
```

If the string will not have an apostrophe, the two forms of the NULL command are equivalent.

You can use the NULL command in a Teradata SQL macro.

**Example 1**

This example shows the results of using the NULL command to specify zero and the hyphen as null characters in two iterations of a select operation:

```
.logon sial/td1,td1database;
defaults
.format on
```
.set null as '0'
select Name,DeptNo,NULLIFZERO(hcap)(Title 'Handicap')from Employee
order by Name;
.set null as '-'
=1;
.format off
.logoff
.exit

BTEQ Response
In response to the first select operation, using zero as the null character, BTEQ returns the following:

*** Query completed. 21 rows found. 3 columns returned.
*** Total elapsed time was 1 second.
90/11/05 select Name ,DeptNo,NULIIFZERO (hcap)(Title 'Handicap ...
Page 1
<table>
<thead>
<tr>
<th>Name</th>
<th>DeptNo</th>
<th>Handicap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aguilar J</td>
<td>600</td>
<td>0</td>
</tr>
<tr>
<td>Brangle B</td>
<td>700</td>
<td>0</td>
</tr>
<tr>
<td>Carter J</td>
<td>500</td>
<td>0</td>
</tr>
<tr>
<td>Chin M</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>Clements D</td>
<td>700</td>
<td>0</td>
</tr>
<tr>
<td>Greene W</td>
<td>100</td>
<td>0</td>
</tr>
</tbody>
</table>

.
.
.
(etc)

In response to the second select operation, using the hyphen as the null character, BTEQ returns the following:

*** Query completed. 21 rows found. 3 columns returned.
*** Total elapsed time was 1 second.
90/11/05 select Name,DeptNo,NULLIFZERO (hcap)(Title 'Handicap ...
Page 1
<table>
<thead>
<tr>
<th>Name</th>
<th>DeptNo</th>
<th>Handicap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aguilar J</td>
<td>600</td>
<td>-</td>
</tr>
<tr>
<td>Brangle B</td>
<td>700</td>
<td>-</td>
</tr>
<tr>
<td>Carter J</td>
<td>500</td>
<td>-</td>
</tr>
<tr>
<td>Chin M</td>
<td>100</td>
<td>-</td>
</tr>
<tr>
<td>Clements D</td>
<td>700</td>
<td>-</td>
</tr>
<tr>
<td>Greene W</td>
<td>100</td>
<td>-</td>
</tr>
</tbody>
</table>

.
.
.
(etc)

Example 2
The Example 1 commands in a Teradata SQL macro appears as:

ECHO '.SET NULL AS ''0'' ';  
ECHO '.SET NULL AS ''-'' ';
ECHO '.SET NULL AS "0"';
ECHO '.SET NULL AS "-"';
Purpose
Excludes specified columns returned from SQL SELECT statements.

Syntax

```
.set defaults
.set format on
.heading "Example 1"
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n )</td>
<td>the number of the columns that you want to omit, ordered from left to right, and separated by a comma or one space.</td>
</tr>
<tr>
<td></td>
<td>The value of ( n ) can range from 1 to 2048.</td>
</tr>
</tbody>
</table>

Usage Notes
The OMIT command is useful to underscore a field or to cause a pagebreak at a field that you do not want to be displayed.

The initial values of the OMIT command are OFF and ALL by default. If you use the command and do not specify ON or OFF, BTEQ assumes ON and ALL.

If you want to produce several reports during the same BTEQ session, and change the OMIT specifications for each report, you must explicitly specify .SET OMIT OFF before supplying each new OMIT command specification.

Turn off any prior OMIT command specifications if you want previously omitted columns to appear on subsequent reports.

You can use the OMIT command in a Teradata SQL macro.

Example 1
The following example shows the results of using the OMIT command specifications three ways:

- With the default configuration
- With all columns omitted
- With one column (column 2) omitted

```sql
database personnel;
.set defaults
.set format on
.heading "Example 1"
```


.set omit off
select deptno,
   Loc(Title 'Department Location')
from department
order by deptno
;
.set format on
.set defaults
.set omit off
.set omit on all
.heading 'Example 2' = 1
.set defaults
.set format on
.set omit off
.set omit on 2
.heading 'Example 3' = 1
.defaults
.set format off
.logoff
.exit

BTEQ Responses

In the first select operation, with the OMIT command specifications set to the default configuration, BTEQ returns the following response:

*** Query completed. 5 rows found. 2 columns returned.

Example 1

<table>
<thead>
<tr>
<th>DeptNo</th>
<th>Department Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>NYC</td>
</tr>
<tr>
<td>300</td>
<td>NYC</td>
</tr>
<tr>
<td>500</td>
<td>ATL</td>
</tr>
<tr>
<td>600</td>
<td>CHI</td>
</tr>
<tr>
<td>700</td>
<td>NYC</td>
</tr>
</tbody>
</table>

In the second select operation, with the OMIT command specifications set to ON ALL, BTEQ returns the following response:

*** Query completed. 5 rows found. 2 columns returned.

Example 2

In the third select operation, with the OMIT command specifications set to omit column two, BTEQ returns the following response:

*** Query completed. 5 rows found. 2 columns returned.

Example 3

<table>
<thead>
<tr>
<th>DeptNo</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
</tr>
<tr>
<td>300</td>
</tr>
<tr>
<td>500</td>
</tr>
<tr>
<td>600</td>
</tr>
<tr>
<td>700</td>
</tr>
</tbody>
</table>
Example 2

The Example 1 command in a Teradata SQL macro appears as:

```sql
ECHO '.SET OMIT ON 2';
```
**Purpose**
Submits an operating system command to a network-attached system.

**Syntax**

```
.OS os_command
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>os_command</code></td>
<td>a string that represents any operating system command that is legal in your operating system</td>
</tr>
</tbody>
</table>

**Usage Notes**
The specified operating system command must be valid for execution on your operating system. To use more than one operating system command, send the BTEQ OS command repeatedly, or send the operating system command that allows control to be given to another program. This is accomplished differently in each operating system.

If you use the OS command to look at the contents of an export file and the file has been closed, you can view all of the file. If you use the OS command to look at the contents of an export file and the file has not been closed, you may not be able to view the last few records in the file, though they transfer to the file when the file is closed. This is because the last few records are still in the buffer, and have not yet been written to disk. (See “EXPORT” on page 152 for more information about closing an export file.)

You can use an OS command in a Teradata SQL macro, but only for one specific command at a time. (That is, you cannot use it to drop into the command processor or shell).

The command will be issued to the operating system in the UTF8 encoding for Workstation BTEQ, whenever the I/O encoding is UTF8 or UTF16. The use of multi-byte characters in the OS command is not supported and may produce unpredictable results.

**Example 1 - List Directory Files**
Use the following OS commands to display the files in your directory:

- For UNIX, specify:
  ```
  OS ls
  ```
- For Windows, specify:
  ```
  OS DIR
  ```
Example 2 - UNIX
For UNIX, use the UNIX command EXECUTE SHELL to exit from BTEQ and press and hold the keys Control and D to return to BTEQ.

For example:
```
.OS exec s
$ pg myfile.one
$ cp oldfile newfile
$ cd draft
$ <cntrl>/d
```

Example 3 - Windows
For Windows, use the DOS command COMMAND (secondary command processor) to exit from BTEQ and the EXIT command to return to BTEQ.

```
.OS command
C> type b:myfile.one
C> edlin b:myfile.one
C> dir a:
C> exit
```

Example 4 - Teradata SQL
In a Teradata SQL macro, the OS command is written as follows:
```
ECHO '.OS ls';
```

Example 5 - Change Window Title
If you have multiple instances of Windows BTEQ active, you can use the .OS command with the DOS command TITLE to customize the console windows’ title text so you can differentiate between them. The following example changes the default title to `mach1-db-code-diagon`:

```
.OS TITLE mach1-db-code-diagon
```
PACK

Purpose
The PACK command provides the capability of sending multiple IMPORT data file records along with an SQL request. BTEQ uses this factor to indicate an upper limit when determining how many records to pack into the USING Data buffer sent with the SQL request.

Syntax

\[
\text{PACK } n
\]

where

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n )</td>
<td>is an integer in the range of 1...999900</td>
</tr>
<tr>
<td>0</td>
<td>means “non specified” which is the default</td>
</tr>
</tbody>
</table>

Usage Notes
The pack factor is reducible to prevent a USING data buffer overflow. BTEQ packs as many records into the request buffers as will fit. To prevent CLI buffer overflow errors, BTEQ considers both the upper limit, indicated by the pack factor, and CLI’s maximum message size. When a pack factor is used, statistics are created for the average and largest factor used as part of the summary messages for the request’s result.

A pack factor is established by using the PACK command or by using a PACK clause for the REPEAT command. See “REPEAT” on page 252 for more information about the REPEAT command.

Note: Large pack factors should be used judiciously. A significant amount of memory is allocated for the associated underlying data pointers, which may require advance planning, particularly for z/OS BTEQ use. Also consider the underlying multi-data-parcel protocol limits imposed by the DBS and/or CLI, which can further constrain the usability of a large pack factor. The PACK factor setting is ignored if the database being used does not support iterated requests.
Purpose
Ejects a page whenever the value for one or more specified columns changes.

Syntax

```
SET PAGEBREAK [ON | OFF] [n | ALL]
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>the number of the columns where you want the page break to occur, ordered from left to right, and separated by commas. The value of n can range from 1 to 2048.</td>
</tr>
</tbody>
</table>

Note: Though using a comma as a separator character between column numbers is the preferred construction, you can use a single space character instead of a comma as a separator character in the BTEQ PAGEBREAK command.

Usage Notes
If you specify more than one column, BTEQ ejects a page when the value of any specified column changes. If a column is not sorted and the column appears in the list of pagebreak column numbers, each line may appear on a separate page.

If you use the FOLDLINE command with the PAGEBREAK command, set the pagebreak column number to break on the first folded line (as set with the FOLDLINE command), as in the following example:

```
.SET PAGEBREAK ON 1;
.SET FOLDLINE ON 1, 2, 3;
```

If the pagebreak column on the PAGEBREAK command does not occur on the first folded line, BTEQ splits the rest of the row onto the next page.

If the PAGEBREAK command is not specified, the values are OFF and ALL by default. If the PAGEBREAK command is used and ON or OFF is not specified, BTEQ assumes ON and ALL.

Note: You must set the FORMAT command option to ON for the PAGEBREAK command to take effect. If you set the PAGEBREAK specification to ON with the FORMAT command option set to OFF, BTEQ ignores the PAGEBREAK command when formatting output, even though you can see its setting in the response to a SHOW CONTROLS command.

You can use the PAGEBREAK command in a Teradata SQL macro.
Example 1
The following example uses the PAGEBREAK command to begin a new page when the value of column 2 changes:

```sql
database personnel;
.set defaults
.set format on
.set pagebreak on 2
.heading "Pagebreak on 2"
select Name, Sex from employee
order by sex;
.defaults
.format off
.logoff
.exit
```

BTEQ Response
BTEQ returns the following two-page response:

**Page one:**

```plaintext
*** Query completed. 21 rows found. 2 columns returned.
Pagebreak on 2
Name    Sex
-------- ---
Newman P F
Smith T  F
Chin M   F
Moffit H F
Regan R  F
Leidner P F
Brangle B F
Marston A F
Phan A   F
```

**Page two:**

```plaintext
Pagebreak on 2
Name    Sex
-------- ---
Peterson J M
Kemper R  M
Inglis C  M
Carter J  M
Aguilar J M
Greene W  M
Watson L  M
Clements D M
Omura H   M
Smith T   M
Russell S M
Reed C    M
```
Example 2
The Example 1 command in a Teradata SQL macro appears as:

```
ECHO `.SET PAGEBREAK ON 2';
```
**PAGELENGTH**

**Purpose**
Specifies the maximum number of lines on a printed page of output.

**Syntax**
```
SET PAGELENGTH n
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>the maximum number of lines on a page.</td>
</tr>
</tbody>
</table>

The value is 55 by default.
Valid range is 1 to 2147483647 (for Workstation).
Valid range is 1 to 2147483407 (for Mainframe).

**Usage Notes**
Use the PAGELENGTH command to customize page length for different reports or forms.

**Note**: You must set the FORMAT command option to ON for the PAGELENGTH command to take effect. If a PAGELENGTH command is used with the FORMAT command option set to OFF, BTEQ ignores the PAGELENGTH command specification when formatting output, even though you can see its setting in the response to a SHOW CONTROLS command.

You can use the PAGELENGTH command in a Teradata SQL macro.

**Example 1**
The following example shows the results of using the PAGELENGTH command to set the maximum page length at 67 lines and at 30 lines for the same select operation:

```sql
database personnel;
.set defaults
.set format on
.heading "Pagelength 67"
.footing "Bottom of Page"
.set pagelength 67
select deptname
,deptno
,loc from department
order by deptname
;
.defaults
.format on
.heading "Pagelength 30"
```
BTEQ Response

In response to the first select operation BTEQ formats the results, prints the footer at line 67, and ejects the page:

```plaintext
*** Query completed. 5 rows found. 3 columns returned.  
Pagenlength 67

DeptName       DeptNo  Loc
--------------  ------  ---
Administration  100    NYC
Engineering     500    ATL
Exec Office     300    NYC
Manufacturing   600    CHI
Marketing       700    NYC
```

(BTEQ counts to line 67)

Example 2

The Example 1 commands in a Teradata SQL macro appears as:

```sql
ECHO '.PAGELENGTH 67';
```

or as:

```sql
ECHO '.PAGELENGTH 30';
```
**QUIET**

**Purpose**
Limits BTEQ output to errors and request processing statistics. BTEQ displays the results in a format that is suitable for performance testing.

**Syntax**

```
SET QUIET ON
SET QUIET OFF
```

**Usage Notes**

The QUIET command is typically used when running performance tests that load data from more than one session.

When you set the QUIET command option to ON, BTEQ suppresses the screen display of:

- Results from an SQL SELECT statement. (If an export file is in effect, the results do reach the export file.)

When you set the QUIET command option to ON, BTEQ limits the screen display of:

- ActivityCount messages. (ActivityCount messages display the number of rows affected.)
  For example:
  ```
  *** Ok, Session 31587, Request 1, Statement# 1
  *** Success, Stmt# 1 ActivityCount = 1
  *** Success, Session 31587, Request 2, Stmt# 1 ActivityCount = 1
  ```
  
  Mainframe BTEQ never suppresses ActivityCount messages. Workstation BTEQ suppresses ActivityCount messages only when single session is used.
  
  - ActivityType messages. (ActivityType messages display how the rows were affected.)
    For example:
    ```
    *** Query completed. One row found. One column returned.
    *** Insert completed. One row added.
    ```
    BTEQ suppresses ActivityType messages if exporting in Field Mode. If not exporting in Field Mode, BTEQ suppresses ActivityType messages only when single session is used.

  - Statistical Messages (Statistical messages display the time elapsed for the submitted request.)
    For example:
    ```
    *** Total elapsed time was 1 second.
    *** Total Query time was 1 second.
    ```
    BTEQ suppresses time elapsed messages if TIMEMSG is set to NONE.
If you use the QUIET command with the REPEAT or = command, BTEQ determines the reporting of time statistics by the number of sessions being run. With multiple sessions, BTEQ only reports the summary times for each cycle (start time, finish time, and total time). During a single session, BTEQ reports summary and processing times of queries executed in each cycle.

Typically, the QUIET and REPEAT commands are used together in multi-session data-load operations.

If you have not specified the QUIET command, the value is OFF by default. If the command is used and ON or OFF is not specified, BTEQ assumes ON.

You can use the QUIET command in a Teradata SQL macro.

**Example 1**
To limit BTEQ output, type:

```
.SET QUIET ON
```

**Example 2**
The Example 1 command in a Teradata SQL macro appears as:

```
ECHO '.SET QUIET ON';
```
QUIT

Purpose
Logs off the Teradata sessions and then exits from BTEQ. If the Teradata sessions are already logged off, the QUIT command just exits from BTEQ.

Syntax

\[ .QUIT \]

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTIVITYCO</td>
<td>the current value of the ACTIVITYCOUNT status value</td>
</tr>
<tr>
<td>ERRORCODE</td>
<td>the error code generated by the last Teradata SQL request as the job step return code</td>
</tr>
<tr>
<td>ERRORLEVEL</td>
<td>the highest error severity level that you assigned</td>
</tr>
<tr>
<td>( n )</td>
<td>a decimal number as the job step return code. To avoid existing codes, do not use numbers from 1 to 10.</td>
</tr>
</tbody>
</table>

Usage Notes
The BTEQ EXIT and QUIT commands are identical. For your own convenience in writing scripts, you may prefer to use the EXIT command after an explicit LOGOFF command and use QUIT if the LOGOFF command is not explicitly provided.

You cannot use the QUIT command in a Teradata SQL macro.

The ERRORCODE value only contains the status of the last statement, even in a set of statements specified with the REPEAT command.

Example
To terminate a program and exit BTEQ, type:

\[ .QUIT \]
**RECORDMODE**

**Purpose**
Returns data from SQL SELECT statements in client-oriented data representations rather than character format.

**Syntax**
```
SET RECORDMODE [ON | OFF]
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFF</td>
<td>Field Mode (system default).</td>
</tr>
<tr>
<td>ON</td>
<td>Record Mode (command default).</td>
</tr>
</tbody>
</table>

**Note:** The BTEQ RECORDMODE command does not affect the data returned from Teradata Database. It specifies only the format of the data.

**Usage Notes**
By default, BTEQ returns data in Field Mode and formats it according to accompanying FORMAT command specifications. In Field Mode, BTEQ returns all data values in character format.

When returning data in Record Mode, BTEQ does not format it. Instead, BTEQ presents each row in the format specified in the Teradata SQL SELECT statement using the representation that is appropriate for the client system. BTEQ usually presents each selected row as a hexadecimal dump, ignoring FORMAT and TITLE command specifications. Null values, data types and data lengths are implicit, and you can use Teradata SQL data conversion to change the format of the data.

Record Mode is automatically set when you specify .EXPORT DATA. The default setting of Field Mode is automatically reset when you specify .EXPORT REPORT or .EXPORT RESET, or when you set either the INDICDATA or RECORDMODE command option to OFF.

If you have not specified the RECORDMODE command, the system value is OFF by default. If you use the command and do not specify ON or OFF, BTEQ assumes ON. Use the .SHOW CONTROLS command to determine what RECORDMODE is set to.

You can use the RECORDMODE command in a Teradata SQL macro.

---

Chapter 5: BTEQ Commands
RECORDMODE
End-of-Line Character Differences

The character string that indicates the end of a line in a BTEQ report differs between UNIX and Windows systems.

<table>
<thead>
<tr>
<th>Operating System</th>
<th>End of Line Indicator</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIX</td>
<td>\n</td>
</tr>
<tr>
<td>Windows</td>
<td>\r\n</td>
</tr>
</tbody>
</table>

This conforms to the standard line ending format for each environment. You will need to alter applications to account for this difference before attempting to run them on both types of environments.

Numeric Overflow in Reports

When in Field Mode, a numeric overflow error returned for numeric or decimal data types in a report, is reported as '***' instead of 'Error.... etc.'.

For example, the following query returns the correct format because the format accommodates the number of digits returned:

```sql
SELECT cast (123456 as integer format '$999,999');
```

The result is:

```
123456
$123,456
```

But the next query returns an overflow reported as stars '***...':

```sql
SELECT CAST (123456 as integer format '$99,999');
```

The result is:

```
123456
******
```

If the same query is executed in Record or Indicator Mode, then the numeric overflow is reported as an error, the preferred response.

Example 1

The following example shows the results of using the RECORDMODE command to return data in client-oriented (Record Mode) representations:

```
database personnel;
.set defaults
.set recordmode on
select deptno ,deptname from department
order by deptno ;
.defaults
.set recordmode off
.logoff
.exit
```
**BTEQ Response**

```plaintext
*** Success, Stmt# 1 ActivityCount = 5
*** Query completed. 5 rows found. 2 columns returned.
*** Record#1. Dump Of Data:
  0000  0064 000E C184 9489 9589 A2A3 9981 A389
  *....Administrati*  0010  9695
  *on*
*** Record#2. Dump Of Data:  0000  012C 000B C5A7 8583
  40D6 8686 8983 85  *....Exec Office*
*** Record#3. Dump Of Data:  0000  01F4 000B C595 8789
  9585 8599 8995 87  *.4..Engineering*
*** Record#4. Dump Of Data:  0000  0258 000D D481 95A4
  8681 83A3 A499 8995 *.4..Manufacturin*
  0010  87
  *g*
*** Record#5. Dump Of Data:  0000  02BC 0009 D481 9992
  85A3 8995 87  *.....Marketing*
```

**Example 2**

The Example 1 command in a Teradata SQL macro appears as:

```
ECHO '.SET RECORDMODE ON';
```

To return data in Field Mode after you have set the RECORDMODE command option to ON, specify another RECORDMODE command as follows:

```
.SET RECORDMODE OFF
```
REMARK

**Purpose**
Places a specified string on the standard output stream.

**Syntax**
```
.REMARK 'string
   //string
   "string"
   "string"
   "string"
   "string"
   "string"
   "string"
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>//</td>
<td>a break between lines.</td>
</tr>
<tr>
<td>string</td>
<td>one line of user-supplied comment text.</td>
</tr>
</tbody>
</table>

**Usage Notes**
Together, up to three strings can contain a maximum of 253 characters, including two pairs of line separator characters, to describe up to 10 lines.

<table>
<thead>
<tr>
<th>System Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>channel-attached</td>
<td>BTEQ prints any additional line separator characters as string text.</td>
</tr>
<tr>
<td>network-attached</td>
<td>BTEQ truncates any additional line separator characters and text.</td>
</tr>
</tbody>
</table>

You should not use special characters within the string because they may be interpreted differently by different output devices. You might have to modify a script that uses special characters if you route your output to another device.

If the string will have an apostrophe (single quote) character, use the second form of the REMARK command (the one with quotes as delimiters) or double the apostrophe. For example:

```
.REMARK "Here's where the two steps merge."
```

or

```
.REMARK 'Here''s where the two steps merge.'
```
If the string will not have an apostrophe, the two forms of the REMARK command are the same.

You can use the REMARK command in a Teradata SQL macro.

**Formatting Remarks**

There are two types of characters to control the format of remarks:

<table>
<thead>
<tr>
<th>Control Characters</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>double slashes (//)</td>
<td>breaks the line, and begins a new line</td>
</tr>
<tr>
<td>solid vertical bars (</td>
<td></td>
</tr>
</tbody>
</table>

**Breaking a Line**

You can break a remark line and begin a new line of remark text by inserting a pair of slashes (/ /) at the desired break point. BTEQ allows up to nine line breaks (10 remark lines maximum) within a REMARK command.

**Separating a Line into Sections**

You can separate a remark line into as many as three sections by inserting a pair of solid vertical lines at the desired separation points. BTEQ allows only three sections (two pairs of vertical bar separators) for each line of the remark.

<table>
<thead>
<tr>
<th>Enter</th>
<th>Which</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>a remark with no separator characters,</td>
<td>leaves the line as one section,</td>
<td>the remark is centered.</td>
</tr>
<tr>
<td>one set of separator characters,</td>
<td>separates the line into two sections,</td>
<td>the first part of the line is left-justified, and the second part of the line is centered.</td>
</tr>
<tr>
<td>two sets of separator characters</td>
<td>separates the line into three sections,</td>
<td>the first part of the line is left-justified, the second part of the line is centered, and the third part of the line is right-justified.</td>
</tr>
</tbody>
</table>

**Note:** You can also use a pair of split vertical bars as a separator character.

**Example 1**

The following examples show the REMARK command with a properly formatted two-line string, and the same string improperly formatted with an additional pair of line separator characters:

```
.REMARK 'This is a line of commentary//' And another line too'

.REMARK 'This is a line //of commentary//
```
And another line//too'

**BTEQ Response**

This is a line of commentary
And another line too
This is a line
of commentary
And another line

**Example 2**

The Example 1 commands in a Teradata SQL macro appear as:

```
ECHO 'REMARK ''This is a line of commentary//
And another line too'' ';

ECHO 'REMARK "This is a line of commentary//
And another line too" ';
```
**Purpose**
Submits the next Teradata SQL request a specified number of times.

**Syntax**

```
.REPEAT  n  *  PACK p
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>n</code></td>
<td>how many times you want to submit the next request. For network-attached systems, if you do not specify a value, the default is 1, which effectively voids the repeat function and submits the next request one time.</td>
</tr>
<tr>
<td><code>*</code></td>
<td>that the next request is to be submitted continuously until the import file runs out of data</td>
</tr>
<tr>
<td><code>RECS r</code></td>
<td>is an integer in the range of 1.2251636603879500</td>
</tr>
<tr>
<td><code>p</code></td>
<td>overrides the SET PACK command setting for the duration of the repeat</td>
</tr>
</tbody>
</table>

**Usage Notes**
The REPEAT command is typically used with Teradata SQL requests that contain a USING clause. Each time the request is submitted, it uses the next data row from the input data stream. The REPEAT command is cancelled if the input file cannot be accessed. The REPEAT command only executes once (for each session) if it is followed by a syntactically incorrect SQL SELECT statement.

The definition of request is very critical to this command. The following sequence, for example, submits only the BT (BEGIN TRANSACTION) statement 10 times, because BTEQ interprets the BEGIN TRANSACTION statement as a complete, single-statement request.

```
.REPEAT 10
BT;
select ... ;
select ... ;
select ... ;
ET;
```
If you enter a sequence like this by mistake, be sure to enter nine extra END TRANSACTION statements. Otherwise, Teradata SQL treats any requests you submit until you log off as part of an unfinished transaction, all of which rolls back when you log off.

To repeat the entire transaction 10 times, use:

```sql
.REPEAT 10
BT
;select ...
;select ...
;select ...
;ET;
```

In this case, because of the placement of the semicolons, BTEQ interprets all of the statements in the transaction as one multi-statement request.

You cannot use the REPEAT command in a Teradata SQL macro.

**PACK Clause**

The syntax of the REPEAT command can be extended to include the PACK clause.

**Usage Notes**

The REPEAT command’s PACK clause overrides the SET PACK command setting for the duration of the repeat. Once the repeat is over, the pack factor returns to the value associated with SET PACK use.

See the PACK command, “PACK” on page 237, for more usage details

**Example 1**

```sql
.REPEAT * PACK 100
```

This equates to

Repeat the request as many times as possible before reaching eof (or max n) and pack up to 100 records with each request. The number of records actually sent is determined at REPEAT completion. The pack factor actually used may vary for each request sent.

**Example 2**

If you need to ensure an exact number of records are transferred, use the RECS clause version for the repeat factor to compensate for any “reduced” requests. For example,

```sql
.REPEAT RECS 200 PACK 100
```

This equates to

Repeat the request as many times as necessary to read up to 200 records and pack a maximum of 100 records with each request.

**Repeat and = Command Differences**

The REPEAT command is similar to the = command, with the following differences:

- You must use the REPEAT command before the request that you want to repeat, and the = command after the request.
The REPEAT command specifies the total number of times that you want to submit the following request; the = command specifies how many more times you want to resubmit the last request.

The REPEAT command can resubmit a request repetitively until the import file is exhausted; the = command cannot.

**REPEAT Error Handling**

Errors encountered within a REPEAT are processed as follows:

<table>
<thead>
<tr>
<th>Error</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>can be re-executed</td>
<td>BTEQ retries the statement, as if it were supplied singly.</td>
</tr>
<tr>
<td>cannot be re-executed</td>
<td>BTEQ proceeds to the next statement and continues processing.</td>
</tr>
</tbody>
</table>

**Using REPEAT When QUIET is ON**

If you use the REPEAT command when the QUIET command option is set to ON and a single session is logged on, BTEQ displays a message indicating the estimated execution time after each statement. For example, if you specify .REPEAT 10, BTEQ returns 10 messages indicating estimated time. However, if multiple sessions are logged on, BTEQ only displays the time the first statement was specified and the time the last statement executed.

**Running Multiple Sessions With REPEAT**

When running multiple sessions with the REPEAT command, specify the REPEAT command with the number of sessions before the Teradata SQL DATABASE statement.

To run three concurrent sessions that repeat the Teradata SQL INSERT request as many times as needed, specify:

```
.SET SESSIONS 3
.LOGON 0/fml,fml
.REPEAT 3
DATABASE Personnel;
.REPEAT *
USING num(SMALLINT)
INSERT INTO Department (DeptNo) VALUES (:num) ;
```

The first REPEAT command specifies the default database for all three sessions. The second REPEAT command runs the three sessions until all data is inserted. If you do not specify the first REPEAT command, only the first session uses the Personnel database. The other two sessions do not have a database specified.

Alternatively, the default database can be specified within the Teradata SQL command that specifies a table name. For example:

```
.SET SESSIONS 3
.LOGON 0/fml,fml
.REPEAT *
USING num(SMALLINT)
INSERT INTO Personnel.Department (DeptNo) VALUES (:num) ;
```
In this example, the default database *Personnel* is used with the table name in the Teradata SQL INSERT statement. This example requires only one REPEAT command.

If deadlocks occur when updating a table with a join index in Teradata mode (using BEGIN TRANSACTION and END TRANSACTION with multiple sessions), ANSI mode with COMMIT is an alternative.

**ANSI Mode**

In ANSI mode, a session opens its transaction by its first SQL statement. The transaction is closed completely by sending either a COMMIT or ROLLBACK statement on the session. If the .SESSION and .REPEAT commands are used in ANSI mode, the COMMIT statement has to be sent along with the repeated SQL statement as one request. For example:

```
.SESSION trans ansi
.SESSIONS 10
.logon TPID/USERID, PASSWD
.import data file = <data file name>
.repeat *
    using i(integer), j(integer)
    insert into <table name> (col1, col2)
    values (:i,:j); COMMIT WORK;
.quit
```

If the repeated request is sent without the COMMIT statement, sooner or later, one of the requests will be blocked by other sessions, and gradually the job will hang due to a deadlock.

**Example 3**

The following example shows a series of SQL SELECT statements used first without a REPEAT (Repeat Off heading), and then repeated twice (Repeat 2 heading):

```
database personnel;
.defaults
.format on
.heading 'Repeat Off'
.select  deptname ,loc ,deptno
.from  department
.order by deptno;
.defaults
.format on
.heading 'Repeat 2'
.repeat 2
.select  deptname ,loc ,deptno
.from  department
.order by deptno;
.format off
.logoff
.exit
```

**BTEQ Response**

```
** Query completed. 5 rows found. 3 columns returned. 
Repeat Off
```
REPEATSTOP

Purpose
When set to ON, REPEATSTOP terminates an entire REPEAT operation if an error occurs that cannot be re-executed. If REPEATSTOP is set to OFF, REPEAT operations are not aborted.

If the REPEATSTOP command has not been used, its value is OFF by default. If you specify the command, but do not specify ON or OFF, BTEQ assumes ON.

Syntax
```
.SET REPEATSTOP
```

Usage Notes

<table>
<thead>
<tr>
<th>REPEATSTOP Setting</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFF</td>
<td>If a non-retryable error occurs, BTEQ rolls back to the previous BEGIN statement, and continues inserting records from the point where the error occurred. This is the default.</td>
</tr>
<tr>
<td>ON</td>
<td>BTEQ terminates the entire REPEAT operation if a non-retryable error occurs</td>
</tr>
</tbody>
</table>

Example
```
.SET REPEATSTOP ON;
```
REPORTALIGN

Purpose
Determines the spacing used for UTF8 report mode output.

Syntax

```
SET REPORTALIGN COMPATIBLE
```

where

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPATIBLE</td>
<td>Displays UTF8 report mode output such that each byte consumes one physical space. If no option is used, this is the default value.</td>
</tr>
<tr>
<td>EQUALWIDTH</td>
<td>Displays UTF8 report mode output such that each character consumes only one physical space.</td>
</tr>
<tr>
<td>COLUMNS</td>
<td>Displays UTF8 report mode output such that each wide and full-width character consumes two physical spaces, while all other characters consume one space. Wide and full-width characters are determined by the Unicode Character Database.</td>
</tr>
</tbody>
</table>

Usage Notes
The REPORTALIGN command only affects true UTF8 sessions for workstation BTEQ. These are sessions where the I/O encoding and the session character set encoding are both UTF8. The REPORTALIGN command will not affect sessions where the I/O is UTF8 and the session character set encoding is UTF16, or vice versa. Similarly, the REPORTALIGN command will not affect interactive sessions using the `-m` command line option.

REPORTALIGN is not supported for mainframe BTEQ.

Depending on the character, UTF8 characters require 1-4 bytes.

Affected Behaviors
Different areas of BTEQ output are affected by the REPORTALIGN command, as shown in the following table.
### Example

**Note:** In the following example, each X represents a 3-byte UTF8 wide character.

```sql
.set reportalign compatible
select ' XXXXX ', ' hello ';

*** Query completed. One row found. 2 columns returned.
*** Total elapsed time was 1 second.

' XXXXX '    ' hello '  
---------------------  ---------------------
XXXXX  hello

.set reportalign equalwidth
select ' XXXXX ', ' hello ';

*** Query completed. One row found. 2 columns returned.
*** Total elapsed time was 1 second.

' XXXXX '    ' hello '  
---------------------  ---------------------
XXXXX  hello

.set reportalign column
select ' XXXXX ', ' hello ';

*** Query completed. One row found. 2 columns returned.
*** Total elapsed time was 1 second.

' XXXXX '    ' hello '  
---------------------  ---------------------
XXXXX  hello
```
RETCANCEL

Purpose
The RETCANCEL command, used in conjunction with the RETLIMIT command, cancels a request when the value specified for rows by the RETLIMIT command is exceeded.

If the RETCANCEL command has not been used, its value is OFF by default. If you use the command but do not specify ON or OFF, BTEQ assumes ON.

Syntax
```
.SET RETCANCEL [ON | OFF]
```

Usage Notes

<table>
<thead>
<tr>
<th>RETCANCEL Setting</th>
<th>With a RETLIMIT Value Specified for Rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFF</td>
<td>BTEQ stops displaying data when the specified limit is exceeded but continues gathering data from Teradata Database. This is the system value by default, and it is usually desirable when executing multi-statement requests because it allows each statement to execute and process records up to the set limit. OFF is the initial value.</td>
</tr>
<tr>
<td>ON</td>
<td>BTEQ cancels the remainder of the Teradata SQL request when the specified limit is reached. ON is assumed when neither ON or OFF is specified.</td>
</tr>
</tbody>
</table>

If you have not specified a RETLIMIT value for rows, the RETCANCEL command does not affect BTEQ.

You can use the RETCANCEL command in a Teradata SQL macro.

Example 1
The following example shows the RETCANCEL command used in conjunction with several RETLIMIT commands:
```
.set retcancel
.set retlimit 1000
.set retlimit 100000000000
*** Warning: Number too large
*** Warning: Extra text found after command . The text '00' not processed (ignored) .
```
Example 2
The Example 1 command in a Teradata SQL macro appears as:

```
ECHO '.SET RETCANCEL ON';
```
RETLIMIT

Purpose
Specifies the maximum number of rows and columns displayed on the terminal or written to a file in response to a Teradata SQL statement.

Syntax

```
SET RETLIMIT n1 n2
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n1</td>
<td>Maximum number of rows to be displayed at a terminal or written to a file. To specify a limit, the value of n1 must be greater than 0 and less than 1000000. To disable the limit function, set n1 = 0.</td>
</tr>
<tr>
<td>n2</td>
<td>Maximum number of columns to be displayed at a terminal or written to a file. This element is optional. Note: You must enter a value for n1 before you can enter a value for n2. The value of n2 can be an asterisk or a number between 1 and 65535. However, you must be logged on to a DBS if the number entered is greater than 2048. An asterisk (*) signifies 2048 number of returnable columns. This value is subject to change to support future releases of Teradata Database. If you want to fix the limit of columns that are displayed, enter a number instead of an asterisk. For the current Teradata Database limit, refer to SQL Fundamentals, Appendix C. The value is 100 by default.</td>
</tr>
</tbody>
</table>

Usage Notes
Teradata Database always returns all of the rows that are selected by a Teradata SQL SELECT statement. By default, however, BTEQ uses a maximum of 100 columns to generate its results. If more than 100 columns are necessary for the results, use the RETLIMIT command to increase the limit for the number of columns retained.

Setting n1 to zero returns all rows selected. In this case, the maximum unlimited numbers of rows is 2,147,483,407 for channel-attached and 2,147,483,647 for network-attached systems.

The RETLIMIT command applies to each Teradata SQL statement, not to Teradata SQL requests. Therefore, if a Teradata SQL request contains several Teradata SQL statements, BTEQ processes the specified number of rows for each statement. When the return limit for a statement is exceeded, BTEQ returns the following messages:
*** RetLimit exceeded.
*** Ignoring the rest of the output for this statement.

Note that using Teradata SQL's TOP operator might be a better choice than the RETLIMIT command when a multi-statement request is comprised of multiple data-returning statements, only one of which is to be constrained. That one statement would employ the TOP operator. When rows are specifically ordered (for example, by an ORDER BY clause), it is the first \( n \) rows that TOP ensures are returned. When the selection is not ordered, TOP returns any \( n \) rows.

You can use the RETLIMIT command in a Teradata SQL macro.

Before you increase the column limit, you may need to reestablish other settings so that the effect of the setting is populated over the additional columns. For example, if you want to omit all but the third column, use the OMIT ON ALL command before you change the column limit. After you change the column limit, use the OMIT ON 3 command. Otherwise, the ON attribute will not be populated for the additional columns.

**Example 1**
The database limit for the maximum number of columns that can be selected sometimes changes. If it is unimportant that this limit can change, and you want to specify the maximum number of rows and columns that should be used for result output, type:

```
.SET RETLIMIT * *
```

**Example 2**
This example specifies the maximum number of rows and columns that should be used for result output when the column retention limit must remain the same, without regard to whether the database limit has been increased:

```
.SET RETLIMIT * 2048;
```

**Example 3**
This example shows a series of SQL SELECT statements used first without a RETLIMIT command (Retlimit Off heading), and then repeated twice with RETLIMIT values of 1 (Retlimit 1 heading), and 4 (Retlimit 4 heading):

```
database personnel;
.defaults
.format on
.heading 'Retlimit off'
select name ,deptno
from employee order by deptno;
.defaults
.format on
.heading "Set Retlimit 1"
.set retlimit 1
=1
.defaults
.format on
.heading "Set Retlimit 4"
.set retlimit 4
=1
```
.format off
.quit

BTEQ Response

*** Query completed. 21 rows found.
2 columns returned.
Retlimit off

<table>
<thead>
<tr>
<th>Name</th>
<th>DeptNo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peterson J</td>
<td>100</td>
</tr>
<tr>
<td>Chin M</td>
<td>100</td>
</tr>
<tr>
<td>Greene W</td>
<td>100</td>
</tr>
<tr>
<td>Moffit H</td>
<td>100</td>
</tr>
<tr>
<td>Russell S</td>
<td>300</td>
</tr>
<tr>
<td>Leidner P</td>
<td>300</td>
</tr>
<tr>
<td>Phan A</td>
<td>300</td>
</tr>
<tr>
<td>Inglis C</td>
<td>500</td>
</tr>
<tr>
<td>Carter J</td>
<td>500</td>
</tr>
<tr>
<td>Watson L</td>
<td>500</td>
</tr>
<tr>
<td>Smith T</td>
<td>500</td>
</tr>
<tr>
<td>Omura H</td>
<td>500</td>
</tr>
<tr>
<td>Reed C</td>
<td>500</td>
</tr>
</tbody>
</table>

Set Retlimit 1

<table>
<thead>
<tr>
<th>Name</th>
<th>DeptNo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peterson J</td>
<td>100</td>
</tr>
</tbody>
</table>

*** Warning: RetLimit exceeded.
*** Ignoring the rest of the output for this statement.
*** Query completed. 21 rows found. 2 columns returned.
Set Retlimit 4

<table>
<thead>
<tr>
<th>Name</th>
<th>DeptNo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peterson J</td>
<td>100</td>
</tr>
<tr>
<td>Chin M</td>
<td>100</td>
</tr>
<tr>
<td>Greene W</td>
<td>100</td>
</tr>
<tr>
<td>Moffit H</td>
<td>100</td>
</tr>
</tbody>
</table>

*** Warning: RetLimit exceeded.
*** Ignoring the rest of the output for this statement.

Example 4
The prior command in a Teradata SQL macro appears as:

ECHO '.SET RETLIMIT 4';

Example 5
To disable the current retention limit for rows, type:

.SET RETLIMIT 0

Note: Setting the RETRYLIMIT value to 0 returns BTEQ to the unlimited return mode.
RETRY

**Purpose**
Resubmits requests that fail under certain operational error conditions.

If the RETRY command has not been used, its value is ON by default. If the RETRY command is used but does not specify ON or OFF, BTEQ assumes ON.

**Syntax**

```
SET RETRY [ON | OFF]
```

**Usage Notes**

When set to ON, the RETRY command option takes effect when BTEQ detects any of the following failure codes:

- 2631 – Transaction aborted due to %VSTR.
- 2639 – Too many simultaneous transactions.
- 2641 – %DBID.%TVMID was restructured. Resubmit.
- 2825 – No record of the last request was found after a Teradata Database restart.
- 2827 – Request was aborted by user or due to statement error.
- 2828 – Request was rolled back during system recovery.
- 2835 – A unique index has been invalidated; resubmit request.
- 3111 – The dispatcher has timed out the transaction.
- 3120 – The request is aborted because of a Teradata Database restart.
- 3231 – Concurrent RDDL changes within a group detected.
- 3319 – TDWM control command timed out.
- 3598 – Concurrent change conflict in database—try again.
- 3603 – Concurrent change conflict in table—try again.
- 3897 – Request aborted due to Teradata Database restart. Resubmit.
- 6699 – Transaction aborted due to replication transport outage.

If any of these error conditions occur, or if an AP Reset condition occurs while a BTEQ job is running, BTEQ:

- Determines which session connections have been lost,
- Issues the appropriate error messages,
- Reconnects the lost sessions, and
- Resubmits any in-progress requests.
Except for the error messages and delayed responses, BTEQ processes the affected requests in a normal fashion, without additional user intervention.

When set to OFF, the RETRY command option will not attempt to reconnect sessions.

Error 2631 – Transaction aborted due to %VSTR, may need to be remapped if the RETRY command option is set to OFF. BTEQ by default assigns Error 2631 an error level of zero. If a request is unsuccessfully retried after a deadlock, the final BTEQ error level will be raised.

The ERRORLEVEL command may be used to remap the default severity (for example, zero for Error 2631). The ERRORMAP command may be used to verify the error level currently assigned to Error 2631.

The RETRY command has no effect on other types of errors that may occur repeatedly while processing a REPEAT or = command, such as error 3807—Database does not exist.

BTEQ does not resubmit a request if Teradata Database restarts when the RETRY command option is set to OFF and the CLOSE option of the EXPORT command is specified. In this case, the query results are lost because the output file is either empty or it contains only some of the returned rows.

On network-attached systems, if the RETRY command option is set to ON during multiple sessions and you use the IMPORT command to provide data for USING modifiers, the data is saved in a buffer. This impacts system memory requirements, especially on a PC. You can calculate the memory requirements by multiplying the number of sessions by the amount of data. For example, if the maximum row size is 4,000 bytes and 10 sessions are in use, then BTEQ uses an additional 40,000 bytes. In this case, to avoid an out-of-memory condition for multiple sessions on the PC, set the RETRY command option to OFF.

When you set the RETRY command option to OFF, it remains off until you either exit BTEQ or use the RETRY command with the ON specification.

You can use the RETRY command in a Teradata SQL macro.

**Example 1**

To disable the RETRY command, type:

```
.SET RETRY OFF
```

**Example 2**

The Example 1 command in a Teradata SQL macro appears as:

```
ECHO '.SET RETRY OFF';
```
RTITLE

Purpose
Specifies a title that appears at the top of each page of a report. Titles specified by the RTITLE command automatically include the date and page number.

Syntax
```
SET RTITLE 'string'
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>text for the title. The string can contain a maximum of 253 characters, including line separator characters, to describe up to 10 lines.</td>
</tr>
</tbody>
</table>

Usage Notes
The RTITLE command and the HEADING command are alternate versions of the same command. They both specify the top ten lines on each page of a report. If you use both commands, the most recent one overrides the earlier one.

The RTITLE command automatically provides the page number, and the date in YY/MM/DD format. (When using the HEADING command you must include the &PAGE and the &DATE substitution values in your header string if you want the page numbers and date to appear in your header.)

You should not use special characters within the string because they may be interpreted differently by different output devices. You might have to modify a script that uses special characters if you route your output to another device.

If the string will have an apostrophe (single quote) character, use the second form of the RTITLE command (the one with quotes as delimiters) or double the apostrophe. For example:

```
.SET RTITLE "December's Results"
```

or

```
.SET RTITLE 'December''s Results'
```

If the string will not have an apostrophe, the two forms of the RTITLE command are the same.
You can use the RTITLE command in a Teradata SQL macro.

**Formatting Headings**
There are two types of characters to control heading formats:

<table>
<thead>
<tr>
<th>Control Characters</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>double slashes (//)</td>
<td>breaks the line, and begins a new line</td>
</tr>
<tr>
<td>solid vertical bars (</td>
<td></td>
</tr>
</tbody>
</table>

**Breaking a Line**
You can break a header line and begin a new line of header text by inserting a pair of slashes (//) at the desired break point. BTEQ allows up to nine line breaks (10 header lines maximum) within a HEADING command.

**Separating a Line into Sections**
You can separate a header line into as many as three sections by inserting a pair of solid vertical lines at the desired separation points. BTEQ allows only three sections (two pairs of vertical bar separators) for each line of the header.

<table>
<thead>
<tr>
<th>Enter</th>
<th>Which</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>a header with no separator characters</td>
<td>leaves the line as one section,</td>
<td>the header is centered.</td>
</tr>
<tr>
<td>one set of separator characters</td>
<td>separates the line into two sections,</td>
<td>the first part of the line is left-justified, and the second part of the line is centered.</td>
</tr>
<tr>
<td>two sets of separator characters</td>
<td>separates the line into three sections,</td>
<td>the first part of the line is left-justified, the second part of the line is centered, and the third part of the line is right-justified.</td>
</tr>
</tbody>
</table>

**Note:** You can also use a pair of split vertical bars as a separator character.

If you specify more than two pairs of vertical bar separator characters, BTEQ does the following:

<table>
<thead>
<tr>
<th>System Type</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>channel-attached</td>
<td>BTEQ interprets the additional characters as header text, and issues a warning that you have exceeded the limit of two pairs of vertical bar separator characters.</td>
</tr>
<tr>
<td>network-attached</td>
<td>BTEQ truncates any additional text including the extra (</td>
</tr>
</tbody>
</table>
For example, in a channel-attached system the following title string:

```
.SET RTITLE '||Confidential Report||Part Two|| Page '
```

returns the following undesirable title:

```
90/08/25  Confidential Report  Part Two||Page 2
```

**Blank Title Lines**

If you enter an RTITLE command with one or more blanks as the title string (for example .SET RTITLE' ' or .SET RTITLE" "), BTEQ returns a blank title.

**Two Apostrophes**

If you use an RTITLE command with two apostrophes or quotation marks with no blanks between them (for example .SET RTITLE " " or .SET RTITLE " "), BTEQ uses as many characters that fit from the SELECT statement as the title string. This is the SELECT statement for which BTEQ is generating a report. Titles, etc. only appear in response to a select.

**Title String Wider than Page Width**

If you specify a title string that is wider than the page width, BTEQ truncates from the right until the title fits on the page. (You can continue a title from one line to the next by including a dash character (-) as the last nonblank character of the line to be continued.)

**Default Headings and Titles**

If you do not use this command to specify a title, or the HEADING command to specify a header, the default one-line title/header provides the date, as many characters of the SELECT statement that fit, and the page number, formatted as follows:

```
&DATE||as many characters of SELECT as will fit||Page&PAGE
```

The SELECT statement in the example refers to the statement for which BTEQ is generating a report. The titles etc. only appear in response to a select.

**Note:** If you set the FORMAT command option to OFF, BTEQ ignores the RTITLE command when formatting output, even though you can see the title specifications in the response to a SHOW CONTROLS command. Always set the FORMAT command option to ON if you want a title to appear in your reports.

A report title set during a session remains in effect until you enter another RTITLE, HEADING, or DEFAULTS command.

**Example 1**

The following example uses *Rtitle On* as the title string:

```
database personnel;
.set defaults on
.set format on
.set rtitle 'Rtitle On'
.select * from department;
.set defaults
.set format off
.logoff
.exit
```
BTEQ Response

*** Query completed. 5 rows found. 4 columns returned.  
90/08/01 Rtitle On Page 1

<table>
<thead>
<tr>
<th>DeptNo</th>
<th>DeptName</th>
<th>Loc</th>
<th>MgrNo</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>Engineering</td>
<td>ATL</td>
<td>10012</td>
</tr>
<tr>
<td>700</td>
<td>Marketing</td>
<td>NYC</td>
<td>10021</td>
</tr>
<tr>
<td>300</td>
<td>Exec Office</td>
<td>NYC</td>
<td>10018</td>
</tr>
<tr>
<td>600</td>
<td>Manufacturing</td>
<td>CHI</td>
<td>10007</td>
</tr>
<tr>
<td>10 0</td>
<td>Administration</td>
<td>NYC</td>
<td>10011</td>
</tr>
</tbody>
</table>

Example 2

The Example 2 command in a Teradata SQL macro appears as:

```sql
ECHO '.SET RTITLE ''a//b//c'' ;
```

or

```sql
ECHO '.SET RTITLE "a//b//c" ;
```
RUN

Purpose
Processes the Teradata SQL requests and BTEQ commands from the specified run file.

Syntax
```
.RUN FILE = filename, SKIP=n
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename</td>
<td>name of the file that contains the BTEQ commands and SQL requests on network-attached systems. If the name of the file includes a comma, semicolon, or blank space character, enclose the entire file name in either single or double quotation marks. Additionally, any time text follows a file name, that file name must be enclosed in quotation marks. File names are case sensitive on systems running under UNIX, and case-insensitive on systems running under Windows or TSO. For channel-attached systems, this is the name of the z/OS JCL DD statement that defines the file from which BTEQ commands are read. An ALLOCATE statement must be used for TSO.</td>
</tr>
<tr>
<td>SKIP = n</td>
<td>number of lines skipped from the beginning of a data set or file before reading BTEQ commands.</td>
</tr>
</tbody>
</table>

Usage Notes
BTEQ supports all QSAM-compatible record formats. The maximum acceptable line length is 254 characters.

If you specify a RUN command within a run file, BTEQ switches and begins reading from the new file; BTEQ does not return to the previous file.

Files created with a RUN command can be chained, but not nested. If a run file contains a RUN command to run itself, BTEQ generates an out-of-memory message after several iterations. Always avoid such a loop.

Note: For information on I/O errors and abends, refer to “I/O Errors and Abends” on page 92.
After exhausting the run file, BTEQ resumes reading commands and requests from the standard input stream.

You cannot use the RUN command in a Teradata SQL macro.

For network-attached systems, a BOM is optional at the beginning of a UTF8 or UTF16 RUN file when a Unicode I/O encoding is being used. And when BTEQ is started with the -m command line option, stdio will be based on the system locale. Therefore, the RUN file must contain locale-specific characters (instead of UTF8/UTF16 encoded characters) and cannot contain a BOM.

For channel-attached systems, a RUN file must be encoded in EBCDIC, even for Unicode sessions.

**Example**

To read commands from the file POSTING, type:

```
.RUN FILE=POSTING
```
Purpose
Specifies the width and contents of the string to be placed between the columns in a report.

Syntax
```
.SET SEPARATOR n
.SET SEPARATOR 'string'
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>number of blanks that separate columns.</td>
</tr>
<tr>
<td></td>
<td>The system value is 2 by default. The maximum value is 254.</td>
</tr>
<tr>
<td>string</td>
<td>string that separates columns.</td>
</tr>
<tr>
<td></td>
<td>The maximum length of the string is 253 characters.</td>
</tr>
</tbody>
</table>

Usage Notes
The .SET SEPARATOR n form of the SEPARATOR command inserts n blanks between columns. The .SET SEPARATOR 'string' form of the command inserts the specified string between columns.

If a separator value prevents a report from fitting on a page, BTEQ truncates the separator on the right until the report fits on the page. BTEQ never truncates the separator to less than one character.

You should not use special characters within the string because they may be interpreted differently by different output devices. You might have to modify a script that uses special characters if you route your output to another device.

If the string will have an apostrophe (single quote) character, use the second form of the SEPARATOR command (the one with quotes as delimiters) or double the apostrophe. For example:
```
.SET SEPARATOR " Item's value "
```

or
```
.SET SEPARATOR ' Item''s value '
```

If the string does not contain an apostrophe, the two forms of the SEPARATOR command are the same.

If you do not specify a value for n, then n = 0.
The optional ALL parameter allows the separators to:

- appear on ALL data lines including the summary lines associated with WITH clauses.
- omit the column title lines if they contain only spaces and separators.

The `.SET SEPARATOR ALL` command should not be used with the SIDETITLES or FOLDLINE command. Also, when the ALL parameter is used with the summary titles appearing to the left side of the data, the summary titles may be over-written by the separators.

You can use the SEPARATOR command in a Teradata SQL macro.

**Example 1**

To specify a vertical bar as the column separator character, type:

```sql
    database Personnel;
    .defaults
    .format on
    .heading "set separator |"
    .set separator "|
    select Name
    ,EmpNo
    ,JobTitle
    from Employee
    order by Name
    ;
    .format off
    .logoff
    .exit
```

**BTEQ Response**

```sql
*** Query completed. 21 rows found. 3 columns returned.  
set separator |
Name   |EmpNo|JobTitle
----------  ----- ------------
Aguilar J |10007|Manager
Brangle B |10020|Salesperson
Carter J  |10016|Engineer
Chin M    |10011|Controller
Clements D |10022|Salesperson
Greene W  |10017|Payroll Ck
Inglis C  |10014|Tech Writer
Kemper R  |10006|Assembler
Leidner F |10003|Secretary
Marston A |10009|Secretary
Moffit H  |10002|Recruiter
.
.
.
(etc)
```

**Example 2**

To specify four blanks as the column separator, type:

```sql
    .SET SEPARATOR 4
```
Example 3
The Example 3 command in a Teradata SQL macro appears as:

   ECHO '.SET SEPARATOR 4';

Example 4
To specify the ALL parameter, type:

   set separator ' | ' all
SESSION CHARSET

**Purpose**
Identifies the name of the character set for the current session.

**Syntax**
```
SET SESSION CHARSET <charstring> | <code>
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>charstring</td>
<td>name of the character set. Using the name, instead of the numeric code, is recommended. The name cannot exceed 30 characters.</td>
</tr>
<tr>
<td>code</td>
<td>numeric code for the character set. Code range: 0 to 255</td>
</tr>
</tbody>
</table>

**Usage Notes**
You can choose the character set either by name or by numeric code. If selecting by name, the value should exist in the DBC.CharTranslations View CharSetName column. If selecting by code, the value should exist in the DBC.CharTranslations View CharSetId column. If you specify an invalid character set name or code, BTEQ displays the resulting error returned from Teradata Database plus the next SQL statement, which is not executed.

**Note:** The Teradata Database collation default on channel-attached systems is EBCDIC; the default for network-attached systems is ASCII.

For more information about these System Table views, refer to *International Character Set Support*.

The -c, -e, and -m command line options may also be used to initially set the session character set and I/O encoding for network-attached systems before any commands/statements are processed. For an interactive Unicode session, it is recommended that the -m option be used along with the -c option. This will allow BTEQ to accept and display locale-specific characters, affecting stdio, RUN files, and MESSAGEOUT files. The session charset will remain UTF8 or UTF 16, which is used for import/export files, stored procedure files, and communication to and from the database.

Examples for using the command line options:
Chapter 5: BTEQ Commands

SESSION CHARSET

For channel-attached systems, command line options are not available. If the “charset_id” within the HSHSPB is not defined, BTEQ will default the session character set and I/O encoding to EBCDIC.

For further information on command line options, see “Command Line Options” on page 37.

If these command line options and “charset_id” within the clispb.dat file are not used, BTEQ will default the session character set and I/O encoding to ASCII.

Note: For channel-attached systems, a “code” should not be used for character sets. Though mainframe CLI may accept code for certain character sets, its support is deprecated, and will not work for Unicode charsets. Names should be used instead, both for HSHSPB entry and for SET SESSION CHARSET command.

Note: You can use the HELP SESSION command to display the current collation and character set of your session. The actual columns returned for the response may vary depending on what Teradata Database release you are using. For a more readable format, use FOLDLINE and SIDETITLES commands before specifying the command.

Note: There are restrictions whenever a UTF16 session is involved for network-attached systems as defined below:

<table>
<thead>
<tr>
<th>Command Line Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bteq</td>
<td>interactive mode; charset and I/O encoding default to ASCII</td>
</tr>
<tr>
<td>bteq -c ASCII &lt; script</td>
<td>batch mode using an ASCII script; charset and I/O encoding are ASCII</td>
</tr>
<tr>
<td>bteq -c KANJISJIS_0S</td>
<td>interactive mode; charset and I/O encoding are KanjiShift-JIS</td>
</tr>
<tr>
<td>bteq -c UTF8 &lt; script</td>
<td>batch mode using a UTF8 script; charset and I/O encoding are UTF8</td>
</tr>
<tr>
<td>bteq -c UTF16 &lt; script</td>
<td>batch mode using a UTF 16 script; charset and I/O encoding are UTF16</td>
</tr>
<tr>
<td>bteq -e UTF8 -c UTF16 &lt; script</td>
<td>batch mode using UTF8 script; charset is UTF16; I/O encoding is UTF8</td>
</tr>
<tr>
<td>bteq -e 62 -c 63 &lt; script</td>
<td>batch mode using a UTF16 script; charset is UTF8; I/O encoding is UTF16</td>
</tr>
<tr>
<td>bteq -c UTF16 -m</td>
<td>interactive mode; charset is UTF16; I/O encoding based on locale</td>
</tr>
<tr>
<td>bteq -c UTF8 -m</td>
<td>interactive mode; charset is UTF8; I/O encoding based on locale</td>
</tr>
</tbody>
</table>
### Chapter 5: BTEQ Commands

#### SESSION CHARSET

**Note:** Avoid switching character sets if it is important to generate all non-Latin characters properly through stdout. Using the SESSION CHARSET command to switch character sets could cause a mixture of incompatible character repertoires depending upon the results coming back from the database.

BTEQ will *not* allow a user to switch to a non-ASCII charset while an EXPORT or MESSAGEOUT file is open.

**Example 1**
To change the character set to German during a session, type:

```
.SET SESSION CHARSET 'GermanASCII'
```

**Example 2**
To return to the default character set, type:

```
.SET SESSION CHARSET
```

**Example 3**
To see the character set established for your session, type:

```
.foldline on all
.sidetitles on
help session;
```

**BTEQ Response**

```text
*** Help information returned. One row.
*** Total elapsed time was 1 second.

User Name DBC
Account Name DBC
Logon Date 05/11/14
Logon Time 10:21:40
Current DataBase DBC
```
Collation ASCII
Character Set ASCII
Transaction Semantics Teradata
Current Date Form Integer Date
Session Time Zone 00:00
Default Character Type LATIN
   Export Latin 1
   Export Unicode 1
   Export Unicode Adjust 0
   Export Kanji SJIS 1
   Export Graphic 0
Default Date Format YY/MM/DD
   Radix Separator .
   Group Separator ,
   Grouping Rule 3
Currency Radix Separator .
Currency Group Separator ,
Currency Grouping Rule 3
   Currency Name US Dollars
      Currency $        
      ISOCurrency USD
Dual Currency Name US Dollars
   Dual Currency $    
   Dual ISOCurrency USD
Default ByteInt format -(3)9
Default Integer format -(10)9
Default SmallInt format -(5)9
Default Numeric format --(I).9(F)
   Default Real format -9.99999999999999E-999
   Default Time format HH:MI:SS.S(F)Z
Default Timestamp format YYYY-MM-DD HH:MI:SS.S(F)Z
Current Role
   Logon Account DBC
   Profile
      LDAP N
Audit Trail Id DBC
Current Isolation Level SR
SESSION RESPBUFLEN

**Purpose**

Overrides the buffer length specified in the `resp_buf_len` entry in CLI’s System Parameter Block (SPB).

**Syntax**

```
SET SESSION RESPBUFLEN
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Buffer Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT</td>
<td>Using the DEFAULT keyword has the same effect as specifying 0, and results in the following values:</td>
</tr>
<tr>
<td></td>
<td>Workstation</td>
</tr>
<tr>
<td></td>
<td>CLI SPB default</td>
</tr>
<tr>
<td></td>
<td>Mainframe</td>
</tr>
<tr>
<td></td>
<td>32767 bytes</td>
</tr>
<tr>
<td>0</td>
<td>Using 0 has the same effect as specifying DEFAULT, and results in the following values:</td>
</tr>
<tr>
<td></td>
<td>Workstation</td>
</tr>
<tr>
<td></td>
<td>CLI SPB default</td>
</tr>
<tr>
<td></td>
<td>Mainframe</td>
</tr>
<tr>
<td></td>
<td>32767 bytes</td>
</tr>
<tr>
<td>MAX32</td>
<td>Specifying MAX32 results in the following values:</td>
</tr>
<tr>
<td></td>
<td>Workstation</td>
</tr>
<tr>
<td></td>
<td>32731 bytes</td>
</tr>
<tr>
<td></td>
<td>Mainframe</td>
</tr>
<tr>
<td></td>
<td>32705 bytes</td>
</tr>
</tbody>
</table>
### Syntax Element | Buffer Length
--- | ---
MAX64 | Specifying MAX64 results in the following values:

- **Workstation**: 65535 bytes
- **Mainframe**: 65473 bytes

MAX1MB | Specifying MAX 1MB results in the following values:

- **Workstation**: 1048418 bytes
- **Mainframe**: 1048418 bytes

256...n | You can specify a number between 256 and $n$ for the buffer length, where $n$ is equivalent to the MAX1MB value for BTEQ.
SESSION SQLFLAG

Purpose
Enables and disables the issue of warnings in response to syntax errors in accordance with Federal Information Processing Standards (FIPS) Publications 127-2 and 127-3 for ANSI compliance.

Syntax
```
.SET SESSION SQLFLAG [ENTRY | INTERMEDIATE | NONE]
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENTRY</td>
<td>A warning is to be issued for syntax errors according to FIPS 127-2 tests.</td>
</tr>
<tr>
<td>INTERMEDIATE</td>
<td>A warning is to be issued for syntax errors according to FIPS 127-3 tests.</td>
</tr>
<tr>
<td>NONE</td>
<td>No warning is to be issued for violations of ANSI-compliant syntax.</td>
</tr>
</tbody>
</table>

Usage Notes
The SQLFLAG option takes effect at logon time. You must be logged off to change the setting of this option. If you try to change the SQLFLAG option while you are logged on, the current setting will remain in effect and BTEQ will display the following warning message:

*** Warning: You must not be logged on to change the SQLFLAG or TRANSACTION settings.

Example 1
To enable warnings in response to syntax errors in accordance with FIPS 127-2, type:
```
.set session SQLFLAG ENTRY
```

Example 2
To disable warnings of violations of ANSI-compliant syntax, specify the SESSION SQLFLAG command as:
```
.set session SQLFLAG NONE;
```
or
```
.set session SQLFLAG;
```

Note: You can omit both the SET and SESSION keywords when using this command.
**BTEQ Response**

Unless BTEQ encounters an error condition, there is no display response to the SESSION SQLFLAG command.
SESSION TRANSACTION

**Purpose**
Specifies whether transaction boundaries are determined by Teradata SQL semantics or ANSI standards.

**Syntax**
```
where:
```

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specifies that Transaction Boundaries will be Determined by</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSI</td>
<td>ANSI standards, with <em>implicit</em> transaction start and <em>explicit</em> transaction end by COMMIT or ROLLBACK.</td>
</tr>
<tr>
<td>BTET</td>
<td>Teradata SQL BEGIN TRANSACTION and END TRANSACTION semantics. BTET mode as described here is also referred to as Teradata mode elsewhere.</td>
</tr>
</tbody>
</table>

**Usage Notes**
Use the SHOW CONTROLS command to see the current setting of the TRANSACTION option. For details, see “SHOW CONTROLS” on page 289.

The TRANSACTION option takes effect at logon time; you must be logged off to change the setting of this option. If you try to change the TRANSACTION option while you are logged on, the current setting will remain in effect and BTEQ will display the following warning message:

*** Warning: You must not be logged on to change the SQLFLAG or TRANSACTION settings.***

After a successful logon, BTEQ displays:

*** Transaction semantics are "%s"***

where the “%s” string is BTET or ANSI, depending on the Teradata Database default value and any specification used with the SESSION TRANSACTION command.

In addition to defining transaction boundaries, setting the TRANSACTION option also affects the following:

- Truncation rules for character string assignments
- Default comparison rules for character strings
- Default of SET or MULTISET for CREATE TABLE statement. The default is SET in Teradata mode, and MULTISET in ANSI mode.
For details on running in ANSI vs. BTET (also referred to as Teradata mode) mode, see SQL Fundamentals.

You cannot use the SESSION TRANSACTION command in a Teradata SQL macro.

**Example 1**
To specify transaction boundaries according to Teradata SQL semantics, type:

```
.set session trans BTET;
```

**Example 2**
To specify transaction boundaries according to ANSI standards, type:

```
.set session trans ANSI:
```

**Note:** You can omit both the SET and SESSION keywords when entering this command.

**BTEQ Response**
Unless BTEQ encounters an error condition, there is no display response to the SESSION TRANSACTION command.
SESSION TWO RESPBUFS

Purpose
Specifies whether CLI double-buffering is used or not.

Syntax

SET SESSION TWO RESPBUFS ON
SET SESSION TWO RESPBUFS OFF

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td>CLI’s double-buffering capabilities will be used. If ON or OFF is not specified in the command, BTEQ assumes ON.</td>
</tr>
<tr>
<td>OFF</td>
<td>CLI’s double-buffering capabilities will not be used.</td>
</tr>
</tbody>
</table>

Usage Notes
The specifications for the SESSION TWO RESPBUFS command always override the two-resp-bufs entries for double-buffering in CLI’s System Parameter Block (SPB). In general, use of double-buffering allows for performance benefits that cannot be realized when using single-buffering. Therefore, BTEQ specifies a default of ON for this setting. For further details about CLI buffering methods, refer to the CLI reference manual.
SESSIONS

Purpose
Specifies the number of Teradata sessions that BTEQ is to log on at the next logon.

Syntax
```
SET SESSIONS n
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>the number of sessions that BTEQ is to run. The maximum value of n is 200. If n is not specified, BTEQ assumes 1.</td>
</tr>
</tbody>
</table>

Note:  The number of sessions may be further limited by the number of sessions currently available from Teradata Database.

For information on session reserve, refer to “DISABLE SESSION RESERVE” and “ENABLE SESSION RESERVE” in Teradata Director Program Reference.

Usage Notes
Specify the SESSIONS command before specifying the LOGON command. The SESSIONS command is not accepted between the time sessions are logged on and the time they are logged off. The most recent SESSIONS command is used for the number of sessions to log on with the next LOGON command.

If you have not used a SESSIONS command since invoking BTEQ, only one session is logged on.

To run more than one session concurrently, use the REPEAT or = command. If you use more than one session, the order in which requests are processed is not guaranteed, all sessions process identical requests, and only the data differs.

When running multiple sessions with the REPEAT command, supply the REPEAT command with the number of sessions before specifying the Teradata SQL DATABASE statement.

Use the SHOW CONTROLS command to determine the current value of n.

You cannot use the SESSIONS command in a Teradata SQL macro.
Example 1
To run three sessions concurrently that repeat the Teradata SQL INSERT request as many times as needed, specify:

```
.SET SESSIONS 3
.LOGON SIA1/KLD
.REPEAT 3
DATABASE Personnel;
.REPEAT *
USING num(SMALLINT)
INSERT INTO Department (DeptNo) VALUES (:num)
;
```

The first REPEAT command specifies the default database for all three sessions. The second REPEAT command runs the three sessions until all the data is inserted. If the first REPEAT command is not specified, only the first session uses the Personnel database. The other two sessions will not have a database set.

Example 2
Or, you can specify the default database within the Teradata SQL command that specifies a table name. For example:

```
.SET SESSIONS 3
.LOGON SIA1/KLD
.REPEAT *
USING num(SMALLINT)
INSERT INTO Personnel.Department (DeptNo) VALUES (:num)
;
```

In this example, the default database Personnel is supplied with the table name on the Teradata SQL INSERT statement. In this case, only one REPEAT command is required.

Example 3
To choose five Teradata Database sessions, type:

```
.SET SESSIONS 5
.LOGON SIA1/KLD
```

BTEQ Response
After supplying your password, BTEQ displays:

```text
*** Logon successfully completed.
*** 5 sessions logged on.
*** Total elapsed time was 7 seconds.
```
## SHOW CONTROLS

### Purpose
Returns the current settings of the formatting command options.

### Syntax
```
SHOW CONTROLS <cmd>
```

### Table

<table>
<thead>
<tr>
<th><code>&lt;cmd&gt;</code></th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHARSET</td>
<td>LOGONPROMPT</td>
</tr>
<tr>
<td>DECIMALDIGITS</td>
<td>MAXERROR</td>
</tr>
<tr>
<td>ECHOREQ</td>
<td>MESSAGEOUT</td>
</tr>
<tr>
<td>ENCRYPTION</td>
<td>NOTIFY</td>
</tr>
<tr>
<td>ERRORLEVEL</td>
<td>NULL</td>
</tr>
<tr>
<td>EXPORT</td>
<td>OMIT</td>
</tr>
<tr>
<td>EXPORTEJECT</td>
<td>PACK</td>
</tr>
<tr>
<td>FOLDLINE</td>
<td>PAGELENGTH</td>
</tr>
<tr>
<td>FOOTING</td>
<td>PREPAREMODE</td>
</tr>
<tr>
<td>FORMAT</td>
<td>PAGEBREAK</td>
</tr>
<tr>
<td>FORMCHAR</td>
<td>QUIET</td>
</tr>
<tr>
<td>FULLYEAR</td>
<td>RECORDMODE</td>
</tr>
<tr>
<td>HEADING</td>
<td>REPEAT</td>
</tr>
<tr>
<td>IMPORT</td>
<td>REPEATSTOP</td>
</tr>
<tr>
<td>INDICDATA</td>
<td>REPORTALIGN</td>
</tr>
<tr>
<td>LARGEDATAMODE</td>
<td>RETCANCELC</td>
</tr>
<tr>
<td>LIMITS</td>
<td>RETLIMIT</td>
</tr>
<tr>
<td>LOGMECH</td>
<td>RETRY</td>
</tr>
<tr>
<td>LOGON</td>
<td>RTITLE</td>
</tr>
<tr>
<td>SESSION CHARSET</td>
<td>SESSION CHARSET</td>
</tr>
<tr>
<td>SESSION RESPBUFFLEN</td>
<td>SESSION RESPBUFSIZE</td>
</tr>
<tr>
<td>SESSION SQLFLAG</td>
<td>SESSION TRANSACTION</td>
</tr>
<tr>
<td>SESSION TRANSACTION</td>
<td>SESSION TWORESPBUFSIZE</td>
</tr>
<tr>
<td>SESSIONS</td>
<td>SIDETITLES</td>
</tr>
<tr>
<td>SKIPDOUBLE</td>
<td></td>
</tr>
<tr>
<td>TDP</td>
<td></td>
</tr>
<tr>
<td>TIMEMSG</td>
<td></td>
</tr>
<tr>
<td>TITLEDASHES</td>
<td></td>
</tr>
<tr>
<td>UNDERLINE</td>
<td></td>
</tr>
<tr>
<td>WIDTH</td>
<td></td>
</tr>
</tbody>
</table>
Usage Notes

Use the SHOW CONTROLS command to verify the BTEQ format settings that users control. (The QUIET command does not suppress the SHOW CONTROLS display.)

Note: The command SHOW, used without any options, displays the last SQL statement.

You can use the SHOW CONTROLS command in a Teradata SQL macro.

Optionally, the SHOW CONTROLS command can be used at the settings level.

For example, SHOW CONTROLS <cmd>; where "cmd" can be any setting that is reflected in the show controls as [SET] <cmd> = <value>.

The keyword “limits” can be used at the settings level with the SHOW CONTROLS command. SHOW CONTROLS lists the predefined defaults, minimums and maximums for applicable settings like Maximum number of sessions or the Maximum request size.

The line “Client Platform Byte Order” in the output of the SHOW CONTROLS command depicts the endianness type of the client machine. The key phrase “byte order” used with the SHOW CONTROLS command displays the output at the settings level.

Example 1

To see the current settings of formatting controls, specify:

.SHOW CONTROLS

BTEQ Response

Channel- and network-attached systems return different information.

Following is an example of the information returned when using BTEQ for z/OS:

```
.show controls ;
  Default Maximum Byte Count       = 1048500
  Default Multiple Maximum Byte Count = 2048
  Current Response Byte Count      = 32725
  Maximum number of sessions       = 200
  Maximum request size             = 32725
  Maximum IMPORT/USING data size   = 32725
  Maximum number of returnable columns= 2048
  Maximum number of WITH clauses   = 10
  Maximum number of times in a REPEAT = 2147483647
  Maximum number of script files   = 30
  Maximum number of lines in a TITLE = 10
  Maximum number of lines per page  = 2147483407
  Maximum string length            = 254
  Maximum number of WIDTH in a report = 65531
  Minimum number of WIDTH in a report = 20

  Maximum Notify MSG text Byte Count  = 254
  Maximum number of bytes saved for
  SUPPRESS, PAGEBREAK, SKIPLINE,
  UNDERLINE, or SKIPDOUBLE commands,
  or for &n substitutions            = 256

  Client Platform Byte Order        = BIG ENDIAN
```
Chapter 5: BTEQ Commands
SHOW CONTROLS

Basic Teradata Query Reference 291

EXPORT  RESET
IMPORT  RECORD
LOGMECH = default
LOGON
REPEAT = 1
RUN
[SET] AUTOKEYRETRIEVE = OFF
[SET] DECIMALDIGITS = 0 (SPB DEFAULT)
[SET] ECHOREQ = ON
[SET] ENCRYPTION = OFF
[SET] ERRORLEVEL = ON
[SET] EXPORTJECT = ON
[SET] FOLDLINE = OFF ALL
[SET] FOOTING = NULL
[SET] FORMAT = ON
[SET] FORMCHAR = DEFAULT
[SET] FULLYEAR = OFF
[SET] HEADING = NULL
[SET] INDICDATA = OFF
[SET] LARGEDATAMODE = OFF
[SET] MAXERROR = OFF
[SET] NOTIFY = OFF
[SET] NULL = ?
[SET] OMIT = OFF ALL
[SET] PACK = 0
[SET] PAGEBREAK = OFF ALL
[SET] PAGELENGTH = 55
[SET] PREPAREMODE = OFF
[SET] QUIET = OFF
[SET] RECORDMODE = OFF
[SET] REPEATSTOP = OFF
[SET] RETCANCEL = OFF
[SET] RETLIMIT = Rows: No Limit Columns: 100
[SET] RETRY = ON
[SET] RTITLE = NULL
[SET] SEPARATOR = two blanks
[SET] SESSION CHARSET = EBCDIC
  import/export encoding = EBCDIC
  stdin/stdout encoding = EBCDIC
[SET] SESSION RESPBUFLEN = 32767
[SET] SESSION SQLFLAG = NONE
[SET] SESSION TRANSACTION = Unknown
[SET] SESSION TOWRESPBUFS = ON
[SET] SESSIONS = 1
[SET] SIDETITLES = OFF for the normal report.
  And, it is ON for results of WITH clause number: 1 2 3 4 5 6 7 8 9 10.
[SET] SKIPDOUBLE = OFF ALL
[SET] SKIPLINE = OFF ALL
[SET] SUPPRESS = OFF ALL
[SET] TOP = NULL
[SET] TIMEMSG = DEFAULT
[SET] TITLEDASHES = ON for the normal report.
  And, it is ON for results of WITH clause number: 1 2 3 4 5 6 7 8 9 10.
[SET] UNDERLINE = OFF ALL
[SET] WIDTH = 75
Resetting Defaults by Logging Off

The LOGON and LOGOFF commands do not reset the format command default values. If you log off from the initial BTEQ session and then use another LOGON command, the resulting sessions inherit the format command values from the prior session. You must use the DEFAULTS command if you want to reset the format commands to their initial default values.

The only times that you can assume the format commands are at the values listed above are when you execute the first LOGON command after invoking BTEQ, and when you use the DEFAULTS command.

The following list shows the initial default condition of BTEQ at the first LOGON. When you use the DEFAULTS command, it resets these formats to the initial default value.

where:

<table>
<thead>
<tr>
<th>Control Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Maximum Byte Count</td>
<td>Default starting size of the request buffer for single-session operations.</td>
</tr>
<tr>
<td>Default Multiple Maximum Byte Count</td>
<td>Default starting size of the request buffer for multiple-session operations.</td>
</tr>
<tr>
<td>Current Response Byte Count</td>
<td>Current size of the request buffer.</td>
</tr>
<tr>
<td>Maximum request size</td>
<td>Absolute maximum size for a request buffer. This is the maximum number of bytes that can be sent to the database for a pre-internal form request.</td>
</tr>
</tbody>
</table>

**Note:** The [SET] Format control elements vary, depending on your system configuration.

**Example 1**

The Example 2 command in a Teradata SQL macro appears as:

```sql
ECHO '.SHOW CONTROLS';
```

Teradata BTEQ 13.10.00.00 for WIN32. Enter your logon or BTEQ command:

```
.show controls ;
Default Maximum Byte Count = 1048500
Default Multiple Maximum Byte Count = 2048
Current Response Byte Count = 32705
Maximum number of sessions = 200
Maximum request size = 32705
Maximum IMPORT/USING data size = 32705
Maximum number of returnable columns= 2048
Maximum number of WITH clauses = 10
Maximum number of times in a REPEAT = 2147483647
Maximum number of script files = 30
Maximum number of lines in a TITLE = 10
Maximum number of lines per file = 2147483647
Maximum string length = 254
Maximum number of WIDTH in a report = 65531
Minimum number of WIDTH in a report = 20
```
Maximum Notify MSG text Byte Count = 254
Maximum number of bytes saved for SUPPRESS, PAGEBREAK, SKIPLINE, UNDERLINE, or SKIPDOUBLE commands, or for &n substitutions = 256

Client Platform Byte Order = LITTLE ENDIAN

MESSAGEOUT FILE=win32.txt
REPEAT = 1
RUN

[SET] AUTOKEYRETRIEVE = OFF
[SET] DECIMALDIGITS = 0 (SPB DEFAULT)
[SET] ECHOREQ = ON
[SET] ENCRYPTION = OFF
[SET] ERRORLEVEL = ON
[SET] FOLDLINE = OFF ALL
[SET] FOOTING = NULL
[SET] FORMAT = OFF
[SET] FORMCHAR = OFF
[SET] FULLYEAR = OFF
[SET] HEADING = NULL
[SET] INDICDATA = OFF
[SET] LARGEDATAMODE = OFF
[SET] LOGONPROMPT = ON
[SET] NOTIFY = OFF
[SET] NULL = ?
[SET] OMIT = OFF ALL
[SET] PACK = 0
[SET] PAGEBREAK = OFF ALL
[SET] PAGELENGTH = 55
[SET] PREPAREMODE = OFF
[SET] QUIT = OFF
[SET] RECORDMODE = OFF
[SET] REPEATSTOP = OFF
[SET] REPORTALIGN = COMPATIBLE
[SET] RETCANCEL = OFF
[SET] RETLIMIT = Rows: No Limit Columns: 100
[SET] RETRY = ON
[SET] RTITLE = NULL
[SET] SEPARATOR = two blanks
[SET] SESSION CHARSET = ASCII
    import/export encoding = ASCII
    stdin/stdout encoding = ASCII
[SET] SESSION RESPBUFSIZE = 8192
[SET] SESSION SQLFLAG = NONE
[SET] SESSION TRANSACTION = Unknown
[SET] SESSION TWORESPBUFS = ON
[SET] SESSIONS = 1
[SET] SIDETITLES = OFF for the normal report.
    And, it is ON for results of WITH clause number: 1 2 3 4 5 6 7 8 9 10.
[SET] SKIPDOUBLE = OFF ALL
[SET] SKIPLINE = OFF ALL
[SET] SUPPRESS = OFF ALL
Chapter 5: BTEQ Commands
SHOW CONTROLS

[SET] TDP               = dbc
[SET] TIMEMSG           = DEFAULT
[SET] TITLEDASHES       = ON for the normal report.
And, it is ON for results of WITH clause number: 1 2 3 4 5 6 7 8 9 10.
[SET] UNDERLINE        = OFF ALL
[SET] WIDTH            = 75
SHOW ERRORMAP

Purpose
Displays the contents of the tables that map error codes to severity levels.

Syntax

```
.SHOW ERRORMAP
```

Usage Notes
Use the SHOW ERRORMAP command when you want to display the return code associated with a specific error. See “ERRORLEVEL” on page 147 for more information.

Data is returned in two sets:

- System Error Mapping
- User Error Mapping

System Error Mapping contains information about all of the default return codes supported by BTEQ and their return values. If you have overridden the severity level of any of these errors with the .SET ERRORLEVEL command, that information is displayed.

User Error Mapping provides the same information for any error codes which are not usually supported, but have been added. If you have not added any new error codes, this section does not display.

Example 1
Use the following command to display the contents of the tables that map error codes to severity levels:

```
.show errormap
.set errorlevel 4155 severity 12
.set errorlevel 4800 severity 8
.show errormap
```

BTEQ Response
The following is an example of the output of the second .SHOW ERRORMAP command:

<table>
<thead>
<tr>
<th>Errorcode</th>
<th>Errorlevel</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
</tr>
<tr>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>12</td>
</tr>
</tbody>
</table>
Example 2

The Example 2 command in a Teradata SQL macro appears as:

```
ECHO '.SHOW ERRORMAP';
```
**SHOW VERSIONS**

**Purpose**
Returns the current level of each BTEQ software module.

**Syntax**
```
.SHOW VERSIONS
```

**Usage Notes**
Use the SHOW VERSIONS command to determine which version of BTEQ is running. The results of this command include associated revision levels for BTEQ's critical software modules that are helpful when reporting BTEQ problems to product support personnel.

**Note:** The command SHOW, used without any options, displays the last SQL statement.

You can use the SHOW VERSIONS command in a Teradata SQL macro.

**Example 1**
To see the revision levels and overall version number for the BTEQ software currently running, invoke BTEQ and then specify:
```
.SHOW VERSIONS
```

**BTEQ Response**
Channel and network-attached systems return different information. The examples that follow can differ from the responses you receive, depending on the operating system and CLI/Data Connector/TDICU/TeraGSS and Database version you are using.

**Network-Attached Systems**
```
.show versions;
BTEQ Version 13b.10.00.00 for Win 32 running Windows Sockets
CLIV2 : 13d.10.00.00
COPERR.H : 13.10.00.01
CLOPTYPES.H : 13.10.00.00
DBCAREA.H : 13.10.00.00
DBCERR.H : 12.01.00.00
DBCHQEP.H : 13.10.00.05
MOSIIF.H : 13.10.00.02
PARCEL.H : 13.10.00.05
TeraGSS : 13.10.01.01
PIOM : 13.10.00.00
TDICU : 13b.10.00.00
Database : 12.00.00.10

BTEQ linking date is Mar 3 2009
```
Channel-Attached Systems

.SHOW VERSIONS
The following is what the results should look similar to
for a z/OS BTEQ build:

.SHOW VERSION;
BTEQ Version 13e.01.00.00 for IBM z/OS

CLIV2 : 13.10.00
COPERR.H : 13.10.00.02
COPTYPES.H : 13.10.00.00
DBCAREA.H : 13.10.00.00
DBCERR.H : 12.00.00.00
DBCHQEP.H : 13.10.00.05
PARCEL.H : 13.10.00.05
TDICU : 13d.10.00.00.00
Database : 12.00.00.10

BTEQ linking date is Jun 4 2009

Example 2
The Example 2 command in a Teradata SQL macro appears as:

ECHO '.SHOW VERSIONS';
**SIDETITLES**

**Purpose**
Positions titles of summary lines on the left side of the data returned by a SELECT statement that included one or more WITH clauses.

**Syntax**

```
.SET SIDETITLES [ON] [OFF] [withlist] [ALL]
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>withlist</td>
<td>A list of numbers, from 1 to 10, separated by commas, that identify the WITH clauses associated with the SIDETITLES command. The number 1 specifies the first WITH clause, 2 specifies the second, and so on. If you specify 0, instead of a withlist number, then side titles are turned off for any part of the response that is not summarized by a WITH clause.</td>
</tr>
</tbody>
</table>

**Usage Notes**
By default, sidetitles are set to OFF for result column headings and ON for summary titles that are specified using the Teradata SQL WITH clause. Thus, a column heading appears above its column and a summary title appears to the left side of the data it describes.

`.SET SIDETITLES ON` overrides these defaults and positions all column headings to the left of their columns.

Typically, `.SET SIDETITLES ON` is used in conjunction with the FOLDLINE command. By setting the FOLDLINE command options to ON ALL and setting the SIDETITLES command option to ON, you can present each field on its own line with its title to the left. This technique can be useful if you have long fields that do not fit the report in any other format.

If you use a WITH clause to subtotal the first column of a result, a summary title appearing to the left of the result is lost. You can retain the summary title by using `.SET SIDETITLES OFF 1` to place it above the summary result.

If you use the SIDETITLES command without choosing ON or OFF, BTEQ assumes ON for both column headings and summary titles.

You can use the SIDETITLES command in a Teradata SQL macro.
Example 1
The following example shows how the SIDETITLES command affects the output:

database personnel;
defaults
format on
heading 'Sidetitles On'
sidetitles on
set suppress on 1
select name,deptno,salary from employee
with sum(salary)(title 'sum of salary') by deptno;
defaults
heading 'Sidetitles Off'
sidetitles off
=1
format off
logoff
.exit

BTEQ Response
*** Query completed. 26 rows found. 3 columns returned.
Sidetitles On
Name  Peterson J  DeptNo 100  Salary  25,000 .00
Name  Chin M  DeptNo 100  Salary  38,000 .00
Name  Greene W  DeptNo 100  Salary  32,500 .00
Name  Moffit H  DeptNo 100  Salary  35,000 .00
----------
sum of salary  130,000 .00
Name  Phan A  DeptNo 300  Salary  55,000 .00
Name  Russell S  DeptNo 300  Salary  65,000 .00
Name  Leidner P  DeptNo 300  Salary  23,000 .00
----------
sum of salary  143,000 .00
Name  Inglis C  DeptNo 500  Salary  34,000 .00
Name  Carter J  DeptNo 500  Salary  44,000 .00
Name  Watson L  DeptNo 500  Salary  56,000 .00
Name  Smith T  DeptNo 500  Salary  42,000 .00
Name  Omura H  DeptNo 500  Salary  40,000 .00
Name  Reed C  DeptNo 500  Salary  30,000 .00
Name  Marston A  DeptNo 500  Salary  22,000 .00
----------
sum of salary  268,000 .00
Name  Kemper R  DeptNo 600  Salary  29,000 .00
Name  Newman P  DeptNo 600  Salary  28,600 .00
Name  Aguilar J  DeptNo 600  Salary  45,000 .00
Name  Regan R  DeptNo 600  Salary  44,000 .00
----------
sum of salary  146,000 .00
Name  Smith T  DeptNo 700  Salary  45,000 .00
Name  Clements D  DeptNo 700  Salary  38,000 .00
Name  Brangle B  DeptNo 700  Salary  30,000 .00
----------
sum of salary  113,000 .00
*** Query completed. 26 rows found. 3 columns returned.
Sidetitles Off
Name  DeptNo  Salary
---  ------  ----------
Peterson J  100  25,000 .00
Chin M  100  38,000 .00
Greene W 100  32,500 .00
Moffit H 100  35,000 .00
----------
sum of salary 130,500 .00

Phan A  300  55,000 .00
Russell S  300  65,000 .00
Leidner P  300  23,000 .00
----------
sum of salary 143,000 .00

Inglis C  500  34,000 .00
Carter J  500  44,000 .00
Watson L  500  56,000 .00
Smith T  500  42,000 .00
Omura H  500  40,000 .00
Reed C  500  30,000 .00
Marston A  500  22,000 .00
----------
sum of salary 268,000 .00

Kemper R  600  29,000 .00
Newman P  600  28,600 .00
Aguilar J  600  45,000 .00
Regan R  600  44,000 .00
----------
sum of salary 146,600 .00

Smith T  700  45,000 .00
Clements D  700  38,000 .00
Brangle B  700  30,000 .00
----------
sum of salary 113,000 .00

**Example 2**

To display fields on separate lines, with titles printed to the left, type:

```
.SET FOLDLINE ON ALL
.SET SIDETITLES ON
```

**Example 3**

The Example 3 commands in a Teradata SQL macro appear as:

```
ECHO '.SET FOLDLINE ON ALL';
ECHO '.SET SIDETITLES ON';
```
**Purpose**

Inserts two blank lines in your report whenever the value of a specified column changes.

**Syntax**

```
SET SKIPDOUBLE n, ALL OFF
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>The column numbers, ordered from left to right, and separated by a comma or one space. The value of n can range from 1 to 2048.</td>
</tr>
</tbody>
</table>

**Usage Notes**

Use the SKIPDOUBLE command to insert blank space between different sections of a report. BTEQ inserts two blank lines when the value of any of the specified columns changes, or if the value of more than one column changes at the same time.

If you specify the ALL option of the SKIPDOUBLE command, BTEQ triple spaces the report. If you specify the ALL option of both the SKIPDOUBLE command and the SKIPLINE command, BTEQ quadruple spaces the report.

The initial configuration of the SKIPDOUBLE command options are OFF and ALL. If you use the command and do not specify ON or OFF, or provide column numbers, BTEQ assumes ON and ALL.

You can use the SKIPDOUBLE command in a Teradata SQL macro.

**Example 1**

In the following example, the first SELECT operation omits the SKIPDOUBLE command and the second SELECT includes it.

```
database personnel;
.defaults
.format on
select name, empno, deptno from employee
order by deptno
;
.defaults
```
Basic Teradata Query Reference

Chapter 5: BTEQ Commands

**Example 2**
To print two blank lines when the value changes for columns 1 or 5, type:

```
.SET SKIPDOUBLE ON 1,5
```

**Example 3**
The Example 3 command in a Teradata SQL macro appears as:

```
ECHO '.SET SKIPDOUBLE ON 1,5';
```
**SKIPLINE**

**Purpose**
Inserts one blank line in your report whenever the value of a specified column changes.

**Syntax**

```
.SET SKIPLINE [ON \n OFF \n ALL \n n]
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>\n</td>
<td>The column numbers, ordered from left to right, and separated by commas.</td>
</tr>
<tr>
<td>\n</td>
<td>The value of \n can range from 1 to 2048.</td>
</tr>
</tbody>
</table>

**Note:** Though using a comma as a separator character between column numbers is the preferred construction, you can use a single space character instead of a comma as a separator character in the BTEQ SKIPLINE command.

**Usage Notes**

Use the SKIPLINE command to insert blank space between different sections of a report. If you specify more than one column, BTEQ inserts a blank line when the value of any of the specified columns changes, or if the value of more than one column changes at the same time.

If you specify the ALL option of the SKIPLINE command, BTEQ double spaces the report. If you specify the ALL option of both the SKIPLINE command and the SKIPDOUBLE command, BTEQ quadruple spaces the report.

The initial configuration of the SKIPLINE command options are OFF and ALL. If you use the command and do not specify ON or OFF, or provide column numbers, BTEQ assumes ON and ALL.

When using the SKIPLINE and SUPPRESS commands together, the same columns that you specify for the SUPPRESS command are specified for the SKIPLINE command. For example, if you specify the following:

```
.set suppress on 1,2,3,4,5
.set skipline on 1,2,3
```

BTEQ sets columns 4 and 5 for the SKIPLINE command. If you want to specify SKIPLINE for columns 1, 2, and 3, enter the following:

```
.set suppress on 1,2,3,4,5
.set skipline off 4,5
```
Also, if you use the SHOW CONTROLS command after using the SUPPRESS command, BTEQ resets the SKIPLINE columns to the same as the SUPPRESS columns.

You can use the SKIPLINE command in a Teradata SQL macro.

Example 1

The following example returns the selected information first with the SKIPLINE command option set to OFF, then to ON for all columns, and then to ON for column 1:

```sql
database personnel;
.defaults
.format on
.heading 'Skipline Off'
select Proj_Id, EmpNo, WkEnd from charges
order by proj_id
;
.defaults
.format on
.heading "Skipline All"
.set skipline all = 1
.defaults
.format on
.heading "Skipline on 1"
.skipline on 1 = 1
.defaults
.format off
.logoff
.exit
```

BTEQ Response

*** Query completed. 24 rows found. 3 columns returned.

<table>
<thead>
<tr>
<th>Project Id</th>
<th>Employee Id</th>
<th>Week Ending</th>
</tr>
</thead>
<tbody>
<tr>
<td>APR-0001</td>
<td>10015</td>
<td>83/02/18</td>
</tr>
<tr>
<td>APR-0002</td>
<td>10010</td>
<td>83/02/18</td>
</tr>
<tr>
<td>APR-0003</td>
<td>10019</td>
<td>83/02/11</td>
</tr>
<tr>
<td>APR-0004</td>
<td>10015</td>
<td>83/02/25</td>
</tr>
<tr>
<td>APR-0005</td>
<td>10010</td>
<td>83/02/18</td>
</tr>
<tr>
<td>APR-0006</td>
<td>10019</td>
<td>83/02/04</td>
</tr>
<tr>
<td>ENG-0002</td>
<td>10004</td>
<td>83/07/29</td>
</tr>
</tbody>
</table>

*** Query completed. 24 rows found. 3 columns returned.

<table>
<thead>
<tr>
<th>Project Id</th>
<th>Employee Id</th>
<th>Week Ending</th>
</tr>
</thead>
<tbody>
<tr>
<td>APR-0001</td>
<td>10015</td>
<td>83/02/18</td>
</tr>
</tbody>
</table>
Example 2

The Example 2 command in a Teradata SQL macro appears as:

```
ECHO '.SET SKIPLINE 2,3';
```
**SUPPRESS**

**Purpose**
Replaces all consecutively repeated values with blank characters in your reports.

**Syntax**
```
.SET SUPPRESS [ ON | OFF ]
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>n</code></td>
<td>the column numbers, ordered from left to right, and separated by commas. The value of <code>n</code> can range from 1 to 2048.</td>
</tr>
</tbody>
</table>

**Usage Notes**
Use the SUPPRESS command to simplify your reports by eliminating all repetitive values from the specified columns.

BTEQ always provides the first occurrence of a new value in each column, regardless of the setting of the SUPPRESS command options. After that, if the SUPPRESS command option is set to ON, BTEQ replaces those values with blank characters in the specified columns. (Each row containing a new value may be preceded by one or more blank lines, depending on the settings of the SKIPLINE and SKIPDOUBLE commands options.)

If a page break occurs between suppressed values, BTEQ presents the value in the first row of the next page and then suppresses the remaining occurrences of that value.

The initial configuration of the SUPPRESS command options are OFF and ALL. If you use the command and do not specify ON or OFF, or provide column numbers, BTEQ assumes ON and ALL.

If you are producing several reports during the same BTEQ session, and you want to use different SUPPRESS command options for each report, you must explicitly specify .SET SUPPRESS OFF ALL before using each new SUPPRESS command.

When using the SUPPRESS and SKIPLINE commands together, the same columns that you specify for the SUPPRESS command are specified for the SKIPLINE command.

For example, if you use the following:
```
.set suppress on 1,2,3,4,5
.set skipline on 1,2,3
```
BTEQ sets columns 4 and 5 for the SKIPLINE command. If you want to specify SKIPLINE for columns 1, 2, and 3, specify the following:

```
.set suppress on 1,2,3,4,5
.set skipline off 4,5
```

Also, if you use the SHOW CONTROLS command after using the SUPPRESS command, BTEQ resets the SKIPLINE columns to the same as the SUPPRESS columns.

You can use the SUPPRESS command in a Teradata SQL macro.

**Example 1**

The following example returns the selected information first with the SUPPRESS command option set to OFF, then to ON for column 1:

```
database personnel;
.defaults
.format on
.heading 'Suppress off'
select  deptno
  ,name  from  employee
order by deptno
;
.defaults
.format on
.heading 'Suppress on 1'
.suppress on 1
  = 1
.format off
.logoff
.exit
```

**BTEQ Response**

```
*** Query completed. 21 rows found. 2 columns returned.
Suppress off

DeptNo  Name
------  ------------
100    Peterson J
100    Chin M
100    Greene W
100    Moffit H
300    Russell S
300    Leidner P
300    Phan A
500    Inglis C
500    Carter J
500    Watson L
500    Smith T
.
(etc)
*** Query completed. 21 rows found. 2 columns returned.
Suppress on 1
DeptNo  Name
------  ------------
100    Peterson J
```
Example 2

To suppress columns 1 and 2 from one report, and then, in the same BTEQ session, suppress column 3 of the next report, use the following SUPPRESS command for the first BTEQ report:

```
.SET SUPPRESS 1, 2
```

Then, use the following SUPPRESS commands to reconfigure the SUPPRESS command options for the next BTEQ report:

```
.SET SUPPRESS OFF ALL
.SET SUPPRESS 3
```

(You must explicitly specify .SET SUPPRESS OFF ALL before using a new SUPPRESS command.)

Example 3

The command in a Teradata SQL macro appears as:

```
ECHO '.SET SUPPRESS 1,2';
```
**Purpose**

Sets your *tdpid* as a default for subsequent logons during the current BTEQ session.

**Syntax**

```
SET TDP tdpid
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>tdpid</td>
<td>Client Type</td>
</tr>
<tr>
<td></td>
<td><em>tdpid</em> Specifies</td>
</tr>
<tr>
<td>network-attached</td>
<td>the network ID of Teradata Database.</td>
</tr>
<tr>
<td></td>
<td>It must be at least one character in length, but not more than 258.</td>
</tr>
<tr>
<td></td>
<td>It may be a full internet name or an IPv4 or IPv6 address.</td>
</tr>
<tr>
<td></td>
<td>For details on entering and IPv4 or IPv6 address in the SET TDP command, see &quot;Entering IPv4 and IPv6 Network Addresses&quot; on page 52.</td>
</tr>
<tr>
<td></td>
<td>The TDP value can be entered in non-quoted form and quoted form.</td>
</tr>
<tr>
<td></td>
<td>A comment can be present before the <em>tdpid</em> value, with or without space separating the comment from the value.</td>
</tr>
<tr>
<td></td>
<td>However, a comment can be accepted after the <em>tdpid</em> value only if the comment and the <em>tdpid</em> are separated by at least one space.</td>
</tr>
</tbody>
</table>

channel-attached | the identifier of the TDP that handles Teradata Database traffic. |
|               | It must be at least one character in length, but not more than 8. |
|               | It may contain any combination of letters and numbers. |
|               | It should begin TDP... Refer to the *Teradata Tools and Utilities Installation Guide for IBM z/OS* for details. |

**Usage Notes**

The name you define in this command must match the TDP identifier already set by your system or database administrator.
The effect of this command is temporary, lasting only as long as the BTEQ session remains open on the client.

**Example 1**

```
.SET TDP quad1
```

**Example 2**

```
.SET TDP /* tdpid */ quad1.td.teradata.com
```

**Example 3**

```
.SET TDP 111.222.333.444 /* IPv4 address */
```

**Example 4**

```
.SET TDP 111.222.333.444:1025 -- IPv4 address with port
```

**Example 5**

```
.SET TDP /* IPv6 address */ [2002:9941:abcd::cebf]
```

**Example 6**

```
.SET TDP /* IPv6 address with port */ [2002:9941:abcd::cebf]:1025
```
**Purpose**

Enables you to print the total time of a specified request.

**Syntax**

```
.SET TIMEMSG [DEFAULT | QUERY | NONE]
```

**Usage Notes**

Use the TIMEMSG DEFAULT command as the default where no Total Time message prints.

Use the TIMEMSG QUERY command to request that an additional Total Query Time message is printed, after all information associated with the request is received.

Use the TIMEMSG NONE command to suppress elapsed time messages for submitted database requests.

**Note:** If you use the combination of SET TIMEMSG QUERY and SET RETLIMIT (for ROWS), the following three time messages will appear:

- The Elapsed Time message which appeared previously.
- An additional Elapsed Time message, which is printed when the RETLIMIT value for rows is engaged.
- The Total Query Time message which is printed at completion.

The Total Query Time message includes the time that is returned in the first Elapsed Time message which appears.

**Example 1**

To print the total time of a specified request, type:

```
.SET TIMEMSG QUERY
```

**Example 2**

The Example 1 command in a Teradata SQL macro appears as:

```
ECHO '.SET TIMEMSG QUERY';
```

**Example 3**

The following example depicts the default for a REPEATED SQL request:

```
.IMPORT DATA FILE = myfile.imp;
.REPEAT 4 ;
```
TIMEMSG

USING ( i integer, c char(1) )
INSERT INTO mytable values (:i,:c);

Output produced ...

+---------+---------+---------+---------+---------+---------+-.
| .IMPORT DATA FILE = myfile.imp;
+---------+---------+---------+---------+---------+---------+-.
| .REPEAT 4 ;
+---------+---------+---------+---------+---------+---------+-.
USING ( i integer, c char(1) )
INSERT INTO mytable values (:i,:c);

*** Starting Row 0 at Wed Jan 17 17:00:35 2007
*** Insert completed. One row added.
*** Total elapsed time was 1 second.
*** Insert completed. One row added.
*** Total elapsed time was 1 second.
*** Insert completed. One row added.
*** Total elapsed time was 1 second.
*** Insert completed. One row added.
*** Total elapsed time was 1 second.
*** Insert completed. One row added.
*** Total elapsed time was 1 second.
*** Finished at input row 4 at Wed Jan 17 17:00:35 2007
*** Total number of statements: 4, Accepted : 4, Rejected : 0
*** Total elapsed time was 1 second.
*** Total requests sent to the DBC = 4
*** Successful requests per second = 4.000

To suppress the generation of elapsed time messages, include the
TIMEMSG

NONE command prior to doing the IMPORT as follows :

.SET TIMEMSG NONE;

.IMPORT DATA FILE = myfile.imp;

.REPEAT 4 ;

USING ( i integer, c char(1) )
INSERT INTO mytable values (:i,:c);

Output with elapsed time messages suppressed ...

+---------+---------+---------+---------+---------+---------+-.
| .SET TIMEMSG NONE;
+---------+---------+---------+---------+---------+---------+-.
| .IMPORT DATA FILE = myfile.imp;
+---------+---------+---------+---------+---------+---------+-.
| .REPEAT 4 ;
+---------+---------+---------+---------+---------+---------+-.
USING ( i integer, c char(1) )
INSERT INTO mytable values (:i,:c);
*** Starting Row 0 at Wed Jan 17 17:00:35 2007

*** Insert completed. One row added.
*** Insert completed. One row added.
*** Insert completed. One row added.
*** Insert completed. One row added.
*** Finished at input row 4 at Wed Jan 17 17:00:35 2007
*** Total number of statements: 4, Accepted : 4, Rejected : 0
*** Total requests sent to the DBC = 4
*** Successful requests per second = 4.000

To further suppress the Activity Count messages, use QUIET ON in addition to TIMEMSG NONE as follows:

.SET QUIET ON;

.SET TIMEMSG NONE;

.IMPORT DATA FILE = myfile.imp;

.REPEAT 4 ;

USING ( i integer, c char(1) )
INSERT INTO mytable values (:i,:c);

Output suppresses elapsed time message along with activity count message ...

+-------------------------------+-------------------------------+-------------------------------+-------------------------------+-------------------------------+-------------------------------+-------------------------------+
.SET QUIET ON;
*** Type QUIET OFF; to resume output.
+-------------------------------+-------------------------------+-------------------------------+-------------------------------+-------------------------------+-------------------------------+-------------------------------+
.SET TIMEMSG NONE;
+-------------------------------+-------------------------------+-------------------------------+-------------------------------+-------------------------------+-------------------------------+-------------------------------+
.IMPORT DATA FILE = myfile.imp;
+-------------------------------+-------------------------------+-------------------------------+-------------------------------+-------------------------------+-------------------------------+-------------------------------+
.REPEAT 4 ;
+-------------------------------+-------------------------------+-------------------------------+-------------------------------+-------------------------------+-------------------------------+-------------------------------+
USING ( i integer, c char(1) )
INSERT INTO mytable values (:i,:c);
*** Starting Row 0 at Wed Jan 17 17:05:29 2007

*** Finished at input row 4 at Wed Jan 17 17:05:29 2007
*** Total number of statements: 4, Accepted : 4, Rejected : 0
*** Total requests sent to the DBC = 4
*** Successful requests per second = 4.000
### Purpose

Enables or inhibits a line of dash characters immediately before a report summary line generated by a WITH clause.

If the TITLEDASHES command is used without specifying ON or OFF, BTEQ assumes ON and 0.

### Syntax

```
SET TITLEDASHES ON OFF

withlist

0

ALL
```

### Syntax Element

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>withlist</td>
<td>A list of numbers, from 1 to 10, separated by commas, that identify the WITH clauses associated with the TITLEDASHES command. The number 1 specifies the first WITH clause, 2 specifies the second, and so on. If you specify 0, instead of a withlist number, then title dashes are turned off for any part of the response that is not summarized by a WITH clause.</td>
</tr>
</tbody>
</table>

### Usage Notes

Use the TITLEDASHES command to enhance the appearance of your reports by adding a row of dash characters before each specified summary line. Each WITH clause in a Teradata SQL SELECT statement produces a summary line. You can use as many as nine WITH clauses in a Teradata SQL SELECT statement, and each WITH clause can have as many as ten expressions.

To enable a row of dash characters before every summary line in a SELECT result, which is the default configuration, specify .SET TITLEDASHES ON without specifying a withlist number. (You can also use this command to restore title dashes for summary lines that have been suppressed by a prior command.)

To enable or inhibit title dashes for a specific summary line, use the TITLEDASHES command with ON or OFF specified for the appropriate withnumber. (If you do not specify a withnumber, the statement applies to data that is not summarized by a WITH clause.)

**Note:** You can also inhibit title dashes for a specific column by entering a null title (for example, `TITLE ''`). To retain title dashes, enter a blank title (for example, `TITLE''`).

You can use the TITLEDASHES command in a Teradata SQL macro.
Example 1
The following example uses the TITLEDASHES command to suppress dashes preceding the summary line specified by the third WITH clause:

```sql
.SET TITLEDASHES OFF 3
.SET HEADING ON
SELECT deptno, empno, salary, edlev
FROM employee
WITH COUNT(empno) BY deptno
WITH SUM(salary) BY deptno
WITH AVG(edlev) BY deptno;
```

BTEQ Response
In the BTEQ output, note the lack of dash characters preceding the summary line for the third column:

```sql
*** Query completed. 36 rows found. 4 columns returned.  
*** Total elapsed time was 4.71 seconds.  
95/05/16 SELECT deptno, empno, salary, edlev from employee Page 1

<table>
<thead>
<tr>
<th>DeptNo</th>
<th>EmpNo</th>
<th>Salary</th>
<th>EdLev</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>10002</td>
<td>35,000.00</td>
<td>18</td>
</tr>
<tr>
<td>100</td>
<td>10017</td>
<td>32,500.00</td>
<td>16</td>
</tr>
<tr>
<td>100</td>
<td>10001</td>
<td>25,000.00</td>
<td>12</td>
</tr>
<tr>
<td>100</td>
<td>10011</td>
<td>38,000.00</td>
<td>16</td>
</tr>
</tbody>
</table>

Count(EmpNo) 4

Sum(Salary) 130,500.00

Average(EdLev) 16

<table>
<thead>
<tr>
<th>DeptNo</th>
<th>EmpNo</th>
<th>Salary</th>
<th>EdLev</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>10018</td>
<td>65,000.00</td>
<td>16</td>
</tr>
<tr>
<td>300</td>
<td>10008</td>
<td>55,000.00</td>
<td>18</td>
</tr>
<tr>
<td>300</td>
<td>10003</td>
<td>23,000.00</td>
<td>16</td>
</tr>
</tbody>
</table>

Count(EmpNo) 3

Sum(Salary) 143,000.00

Average(EdLev) 17

<table>
<thead>
<tr>
<th>DeptNo</th>
<th>EmpNo</th>
<th>Salary</th>
<th>EdLev</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>10009</td>
<td>22,000.00</td>
<td>18</td>
</tr>
<tr>
<td>500</td>
<td>10016</td>
<td>44,000.00</td>
<td>20</td>
</tr>
<tr>
<td>500</td>
<td>10010</td>
<td>30,000.00</td>
<td>16</td>
</tr>
<tr>
<td>500</td>
<td>10014</td>
<td>34,000.00</td>
<td>16</td>
</tr>
<tr>
<td>500</td>
<td>10015</td>
<td>40,000.00</td>
<td>16</td>
</tr>
<tr>
<td>500</td>
<td>10004</td>
<td>42,000.00</td>
<td>18</td>
</tr>
<tr>
<td>500</td>
<td>10012</td>
<td>56,000.00</td>
<td>20</td>
</tr>
</tbody>
</table>

Count(EmpNo) 7

Sum(Salary) 268,000.00
Example 2

The Example 2 command and request in a Teradata SQL macro appear as:

```
ECHO '.SET TITLEDASHES OFF 3';
SELECT deptno, empno, salary, edlev
FROM employee
WITH COUNT(empno) BY deptno
WITH SUM(salary) BY deptno
WITH AVG(edlev) BY deptno;
```
Purpose
Submits a TSO command in z/OS TSO only. Not valid on network-attached systems.

Syntax

```
.TSO tso_command
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tso_command</code></td>
<td>Any TSO command or program</td>
</tr>
</tbody>
</table>

Usage Notes
Use the TSO command to allocate files, list catalogs, or perform other TSO activities. The TSO command requires an interactive TSO environment. The command cannot be run with a z/OS batch, whether you are using the TSO terminal monitor program (TMP) or not.

You can use the TSO command in a Teradata SQL macro, but only for one command at a time.

Example 1
To allocate a file named ABC, specify:

```
.TSO ALLOC DDNAME(ABC) DSNAME(ABC) SHR
```

Example 2
The Example 1 command in a Teradata SQL macro appears as:

```
ECHO '.TSO ALLOC DDNAME';
```
UNDERLINE

**Purpose**

Inserts a dashed line across the entire width of a report whenever the value of one or more specified columns changes.

**Syntax**

```
SET UNDERLINE [ON | OFF] [ALL | n]
```

where:

<table>
<thead>
<tr>
<th>Syntax Element</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>n</code></td>
<td>The column numbers, ordered from left to right, and separated by a comma or one space.</td>
</tr>
<tr>
<td></td>
<td>The value of <code>n</code> can range from 1 to 2048.</td>
</tr>
</tbody>
</table>

**Usage Notes**

Use the UNDERLINE command to improve the appearance of your reports by separating the data into logical sections. When the value of a specified column changes, BTEQ inserts a dashed line across the entire width of the report, not just across the width of the column specified.

If you specify more than one column, BTEQ inserts a dashed line when the value of any of the specified columns changes, or if the value of more than one column changes at the same time.

The initial configuration of the UNDERLINE command options are OFF and ALL. If you use the command and do not specify ON or OFF, or provide column numbers, BTEQ assumes ON and ALL.

On network-attached systems, the terminal is not treated as an intelligent terminal, so the underline is implemented by drawing a line on the next line below the line of text.

You can use the UNDERLINE command in a Teradata SQL macro.

**Example 1**

The following example returns the selected information first with the UNDERLINE command option set to OFF, then to ON for all columns:

```
database personnel;
.defaults
.format on
.heading 'Underline Off'
select Proj_id
```
Chapter 5: BTEQ Commands

BTEQ Response

*** Query completed. 24 rows found. 3 columns returned

Underline Off

<table>
<thead>
<tr>
<th>Project Id</th>
<th>Project Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP1-0001</td>
<td>A/P Payable DB Design</td>
<td>83/04/10</td>
</tr>
<tr>
<td>AP1-0002</td>
<td>A/P Payable Online System</td>
<td>83/04/21</td>
</tr>
<tr>
<td>AP1-0003</td>
<td>A/P Payable Batch System</td>
<td>83/04/21</td>
</tr>
<tr>
<td>AP2-0001</td>
<td>A/P Payable DB Design</td>
<td>83/04/10</td>
</tr>
<tr>
<td>AP2-0002</td>
<td>A/P Payable Online System</td>
<td>83/04/10</td>
</tr>
<tr>
<td>AP2-0003</td>
<td>A/P Payable Batch System</td>
<td>83/04/10</td>
</tr>
<tr>
<td>AR1-0001</td>
<td>A/R RECV database Design</td>
<td>83/04/21</td>
</tr>
<tr>
<td>AR1-0002</td>
<td>A/R RECV Online System</td>
<td>83/04/10</td>
</tr>
</tbody>
</table>

(etc)
Example 2
To insert a dashed line across the page whenever columns 3 and 7 change value, type:

```
.SET UNDERLINE ON 3,7
```

Example 3
The Example 3 command in a Teradata SQL macro appears as:

```
ECHO '.SET UNDERLINE ON 3,7';
```
### WIDTH

**Purpose**
Specifies the maximum number of characters for each line in your report.

**Syntax**

```
SET WIDTH n
```

where:

**Syntax Element** | **Specification**
---|---
`n` | The page width in characters. The value for data exported to a file in report format is 254 characters by default. The minimum allowed width is 20 characters; the maximum is 65531 characters.

**Usage Notes**
The number that you specify with the WIDTH command determines:

- The maximum number of characters in a line of the report that is either printed or displayed at your terminal.
- The record length of the report file. (The WIDTH specification must be equal to or less than the LRECL. The report lines will be truncated if the WIDTH specification is greater than the LRECL. For Unicode UTF8/UTF16 sessions, LRECL must allow for up to 4 bytes per character of the WIDTH.)

**Note:** The WIDTH specification is the total count of printable characters; it does not include the carriage control characters.

**Note:** Carriage control for EXPORT REPORT/DIF files is not supported for Unicode on BTEQ running under z/OS.

**Note:** You can use the WIDTH command after specifying the EXPORT command to change the WIDTH of the exported file. Upon EXPORT RESET, WIDTH is reset to its original value, the value that was in effect before that EXPORT.

The value of the WIDTH command option when BTEQ is invoked is 75 characters by default.

You can use the WIDTH command in a Teradata SQL macro.

For channel-attached systems, setting a variable RECFM (V, VA, VB, VBA) is recommended whenever a multi-byte session character set is used, since a single character could be multiple bytes.
DBS Export Width

When using BTEQ to format non-UTF16 query results containing multi-byte characters, a BTEQ WIDTH setting (or the LRECL for the SYSPRINT dataset) should be specified which takes the rules for DBS Export Width into consideration. If the BTEQ WIDTH setting is too small, there is a possibility that the bytes for a particular character can get truncated. For example, if a multi-byte string is truncated or wrapped to the next line of a report because a line is too long, the SI character may be lost, or a multi-byte character may be broken. See the International Character Set Support reference for further details on DBS Export Width.

Example 1

The following example returns the selected information first with the WIDTH specification set at the screen default, 75 characters, then with it set at 40 and 20 characters:

database personnel;
.defaults
.format on
.set foldline on all
.set sidetitles on
.skipline on 4
.heading 'Width Off'
.select deptname (title 'Department')
,loc (title 'Loc')
,mgrno (title 'Manager No')
.from department
.order by loc
;
.defaults
.format on
.set foldline on
.set sidetitles on
.heading 'width 40'
.width 40
= 1
.defaults
.format on
.set foldline on
.sidetitles
.heading 'width 20'
.width 20
= 1
.defaults
.format off
.logoff
.exit

BTEQ Response

*** Query completed. 5 rows found. 3 columns returned.  

Width Off

<table>
<thead>
<tr>
<th>Department</th>
<th>Engineering</th>
<th>Loc</th>
<th>ATL</th>
<th>Manager No</th>
<th>10012</th>
</tr>
</thead>
<tbody>
<tr>
<td>Department</td>
<td>Manufacturing</td>
<td>Loc</td>
<td>CHI</td>
<td>Manager No</td>
<td>10007</td>
</tr>
<tr>
<td>Department</td>
<td>Marketing</td>
<td>Loc</td>
<td>NYC</td>
<td>Manager No</td>
<td>10021</td>
</tr>
<tr>
<td>Department</td>
<td>Administrat10n</td>
<td>Loc</td>
<td>NYC</td>
<td>Manager No</td>
<td>10011</td>
</tr>
<tr>
<td>Department</td>
<td>Exec Office</td>
<td>Loc</td>
<td>NYC</td>
<td>Manager No</td>
<td>10018</td>
</tr>
</tbody>
</table>

*** Query completed. 5 rows found. 3 columns returned.
<table>
<thead>
<tr>
<th>Department</th>
<th>Location</th>
<th>Manager No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Engineering</td>
<td>ATL</td>
<td>10012</td>
</tr>
<tr>
<td>Manufacturing</td>
<td>CHI</td>
<td>10007</td>
</tr>
<tr>
<td>Marketing</td>
<td>NYC</td>
<td>10021</td>
</tr>
<tr>
<td>Administration</td>
<td>NYC</td>
<td>10011</td>
</tr>
<tr>
<td>Exec Office</td>
<td>NYC</td>
<td>10018</td>
</tr>
</tbody>
</table>

*** Query completed. 5 rows found. 3 columns returned.

If you enter an invalid WIDTH specification, BTEQ displays an error message as shown in the following example:

```
.set width 65532
***Error: Width out of legal range of 20 to 65531.
```

**Example 2**

The Example 2 command in a Teradata SQL macro appears as:

```
ECHO '.SET WIDTH 60';
```
How to Read Syntax Diagrams

Syntax Diagram Conventions

Notation Conventions

<table>
<thead>
<tr>
<th>Item</th>
<th>Definition / Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Letter</td>
<td>An uppercase or lowercase alphabetic character ranging from A through Z.</td>
</tr>
<tr>
<td>Number</td>
<td>A digit ranging from 0 through 9.</td>
</tr>
<tr>
<td></td>
<td>Do not use commas when typing a number with more than 3 digits.</td>
</tr>
<tr>
<td>Word</td>
<td>Variables and reserved words.</td>
</tr>
<tr>
<td></td>
<td>• UPPERCASE LETTERS represent a keyword.</td>
</tr>
<tr>
<td></td>
<td>Syntax diagrams show all keywords in uppercase, unless operating system restrictions</td>
</tr>
<tr>
<td></td>
<td>require them to be in lowercase.</td>
</tr>
<tr>
<td></td>
<td>• lowercase letters represent a keyword that you must type in lowercase, such as a</td>
</tr>
<tr>
<td></td>
<td>UNIX command.</td>
</tr>
<tr>
<td></td>
<td>• lowercase italic letters represent a variable such as a column or table name.</td>
</tr>
<tr>
<td></td>
<td>Substitute the variable with a proper value.</td>
</tr>
<tr>
<td></td>
<td>• lowercase bold letters represent a variable that is defined immediately following</td>
</tr>
<tr>
<td></td>
<td>the diagram that contains the variable.</td>
</tr>
<tr>
<td></td>
<td>• UNDERLINEDED LETTERS represent the default value.</td>
</tr>
<tr>
<td></td>
<td>This applies to both uppercase and lowercase words.</td>
</tr>
<tr>
<td>Spaces</td>
<td>Use one space between items such as keywords or variables.</td>
</tr>
<tr>
<td>Punctuation</td>
<td>Type all punctuation exactly as it appears in the diagram.</td>
</tr>
</tbody>
</table>

Paths

The main path along the syntax diagram begins at the left with a keyword, and proceeds, left to right, to the vertical bar, which marks the end of the diagram. Paths that do not have an arrow or a vertical bar only show portions of the syntax.

The only part of a path that reads from right to left is a loop.
Continuation Links
Paths that are too long for one line use continuation links. Continuation links are circled letters indicating the beginning and end of a link:

When you see a circled letter in a syntax diagram, go to the corresponding circled letter and continue reading.

Required Entries
Required entries appear on the main path:

If you can choose from more than one entry, the choices appear vertically, in a stack. The first entry appears on the main path:

Optional Entries
You may choose to include or disregard optional entries. Optional entries appear below the main path:

If you can optionally choose from more than one entry, all the choices appear below the main path:
Syntax Diagram Conventions

Some commands and statements treat one of the optional choices as a default value. This value is **UNDERLINED**. It is presumed to be selected if you type the command or statement without specifying one of the options.

**Strings**

Strings appear in single quotes:

```
--------------- 'msgtext' --------------
```

If the string text includes a single quote or a blank space, the string appears in double quotes:

```
--------------- "abc'd" --------------
--------------- "abc d" --------------
```

**Abbreviations**

If a keyword or a reserved word has a valid abbreviation, the unabbreviated form always appears on the main path. The shortest valid abbreviation appears beneath.

```
SHOW CONTROLS
```

In the above syntax, the following formats are valid:

- SHOW CONTROLS
- SHOW CONTROL

**Loops**

A loop is an entry or a group of entries that you can repeat one or more times. Syntax diagrams show loops as a return path above the main path, over the item or items that you can repeat:
Appendix A: How to Read Syntax Diagrams
Syntax Diagram Conventions

Read loops from right to left.

The following conventions apply to loops:

<table>
<thead>
<tr>
<th>Loop Convention</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A maximum number of entries is allowed.</td>
<td>The number appears in a circle on the return path.</td>
</tr>
<tr>
<td></td>
<td>In the example, you may type <code>cname</code> a maximum of 4 times.</td>
</tr>
<tr>
<td>A minimum number of entries is required.</td>
<td>The number appears in a square on the return path.</td>
</tr>
<tr>
<td></td>
<td>In the example, you must type at least three groups of column names.</td>
</tr>
<tr>
<td>A separator character is required between entries.</td>
<td>The character appears on the return path.</td>
</tr>
<tr>
<td></td>
<td>If the diagram does not show a separator character, use one blank space.</td>
</tr>
<tr>
<td></td>
<td>In the example, the separator character is a comma.</td>
</tr>
<tr>
<td>A delimiter character is required around entries.</td>
<td>The beginning and end characters appear outside the return path.</td>
</tr>
<tr>
<td></td>
<td>Generally, a space is not needed between delimiter characters and entries.</td>
</tr>
<tr>
<td></td>
<td>In the example, the delimiter characters are the left and right parentheses.</td>
</tr>
</tbody>
</table>

Excerpts

Sometimes a piece of a syntax phrase is too large to fit into the diagram. Such a phrase is indicated by a break in the path, marked by `(|)` terminators on either side of the break. The name for the excerpted piece appears between the terminators in boldface type.
The boldface excerpt name and the excerpted phrase appears immediately after the main diagram. The excerpted phrase starts and ends with a plain horizontal line:

Multiple Legitimate Phrases

In a syntax diagram, it is possible for any number of phrases to be legitimate:

In this example, any of the following phrases are legitimate:

- dbname
- DATABASE dbname
- tname
- TABLE tname
- vname
- VIEW vname
Appendix A: How to Read Syntax Diagrams
Syntax Diagram Conventions

Sample Syntax Diagram

```
CREATE VIEW viewname AS
  (cname)
  AS
  LOCKING
  LOCK

DATABASE dbname
  TABLE tname
  VIEW vname

FOR IN

ACCESS
  SHARE
  READ
  WRITE
  EXCLUSIVE
  EXCL
  MODE

SELECT expr,
  tname,
  qual_cond
  FROM name
  .aname

HAVING cond

WHERE cond
  GROUP BY
cname
  ,
  col_pos

;}

Diagram Identifier

The alphanumeric string that appears in the lower right corner of every diagram is an internal identifier used to catalog the diagram. The text never refers to this string.
Many examples in this manual refer to a sample database called Personnel. This appendix describes the Personnel database and how to load it onto your system.

Database Contents

The Personnel database has four tables:

- Employee table, shown in Table 10
- Department table, shown in Table 11
- Project table, shown in Table 12
- Charges table, shown in Table 13

Table 10: Employee Table, Personnel Database

<table>
<thead>
<tr>
<th>EmpNo</th>
<th>Name</th>
<th>DeptNo</th>
<th>JobTitle</th>
<th>Salary</th>
<th>Yrs Exp</th>
<th>DOB</th>
<th>Sex</th>
<th>Race</th>
<th>Mstate</th>
<th>EdLev</th>
<th>HCap</th>
</tr>
</thead>
<tbody>
<tr>
<td>10019</td>
<td>Newman P</td>
<td>600</td>
<td>Test Tech</td>
<td>28,600.00</td>
<td>6</td>
<td>Aug 29, 1956</td>
<td>F</td>
<td>C</td>
<td>M</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>10011</td>
<td>Chin M</td>
<td>100</td>
<td>Controller</td>
<td>38,000.00</td>
<td>11</td>
<td>Nov 29, 1955</td>
<td>F</td>
<td>A</td>
<td>M</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>10007</td>
<td>Agular J</td>
<td>600</td>
<td>Manager</td>
<td>45,000.00</td>
<td>11</td>
<td>Jul 09, 1949</td>
<td>M</td>
<td>S</td>
<td>M</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>10018</td>
<td>Russell S</td>
<td>300</td>
<td>President</td>
<td>65,000.00</td>
<td>25</td>
<td>Jun 05, 1932</td>
<td>M</td>
<td>B</td>
<td>D</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>10022</td>
<td>Clements D</td>
<td>700</td>
<td>Salesperson</td>
<td>38,000.00</td>
<td>9</td>
<td>Aug 23, 1944</td>
<td>M</td>
<td>C</td>
<td>M</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>10006</td>
<td>Kemper R</td>
<td>600</td>
<td>Assembler</td>
<td>29,000.00</td>
<td>7</td>
<td>Sep 12, 1947</td>
<td>M</td>
<td>C</td>
<td>M</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>10014</td>
<td>Inglis C</td>
<td>500</td>
<td>Tech Writer</td>
<td>34,000.00</td>
<td>5</td>
<td>Mar 07, 1938</td>
<td>M</td>
<td>C</td>
<td>S</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>10003</td>
<td>Leidner P</td>
<td>300</td>
<td>Secretary</td>
<td>23,000.00</td>
<td>13</td>
<td>Jul 12, 1948</td>
<td>F</td>
<td>C</td>
<td>M</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>10021</td>
<td>Smith T</td>
<td>700</td>
<td>Manager</td>
<td>45,000.00</td>
<td>10</td>
<td>Jul 29, 1946</td>
<td>F</td>
<td>B</td>
<td>S</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>10012</td>
<td>Watson L</td>
<td>500</td>
<td>Vice Pres</td>
<td>56,000.00</td>
<td>8</td>
<td>Oct 03, 1943</td>
<td>M</td>
<td>C</td>
<td>S</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>10004</td>
<td>Smith T</td>
<td>500</td>
<td>Engineer</td>
<td>42,000.00</td>
<td>10</td>
<td>Oct 31, 1951</td>
<td>M</td>
<td>C</td>
<td>M</td>
<td>18</td>
<td>0</td>
</tr>
<tr>
<td>10016</td>
<td>Carter J</td>
<td>500</td>
<td>Engineer</td>
<td>44,000.00</td>
<td>20</td>
<td>Mar 12, 1935</td>
<td>M</td>
<td>C</td>
<td>M</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>10008</td>
<td>Phan A</td>
<td>300</td>
<td>Vice Pres</td>
<td>55,000.00</td>
<td>12</td>
<td>Jun 07, 1947</td>
<td>F</td>
<td>A</td>
<td>M</td>
<td>18</td>
<td>0</td>
</tr>
<tr>
<td>10013</td>
<td>Regan R</td>
<td>600</td>
<td>Purchaser</td>
<td>44,000.00</td>
<td>10</td>
<td>Oct 20, 1948</td>
<td>F</td>
<td>C</td>
<td>M</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>10017</td>
<td>Greene W</td>
<td>100</td>
<td>Payroll Ck</td>
<td>32,500.00</td>
<td>15</td>
<td>Nov 27, 1955</td>
<td>M</td>
<td>N</td>
<td>M</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>10009</td>
<td>Marston A</td>
<td>500</td>
<td>Secretary</td>
<td>22,000.00</td>
<td>12</td>
<td>Jun 07, 1947</td>
<td>F</td>
<td>A</td>
<td>M</td>
<td>18</td>
<td>0</td>
</tr>
</tbody>
</table>
### Appendix B: Loading and Using the Sample Personnel Database

#### Database Contents

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Title</th>
<th>Salary</th>
<th>CDate</th>
<th>Gender</th>
<th>Race</th>
<th>Age</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>10002</td>
<td>Moffit H</td>
<td>Recruiter</td>
<td>35,000.00</td>
<td>Nov 16, 1945</td>
<td>F</td>
<td>B</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>10010</td>
<td>Reed C</td>
<td>Technician</td>
<td>30,000.00</td>
<td>Apr 08, 1949</td>
<td>M</td>
<td>C</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>10015</td>
<td>Omura H</td>
<td>Programmer</td>
<td>40,000.00</td>
<td>Apr 24, 1954</td>
<td>M</td>
<td>A</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>10020</td>
<td>Brangle B</td>
<td>Salesperson</td>
<td>30,000.00</td>
<td>Oct 15, 1947</td>
<td>F</td>
<td>C</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>10001</td>
<td>Peterson J</td>
<td>Payroll Ck</td>
<td>25,000.00</td>
<td>Mar 27, 1942</td>
<td>M</td>
<td>C</td>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>

#### Table 10: Employee Table, Personnel Database (continued)

<table>
<thead>
<tr>
<th>DeptNo</th>
<th>DeptName</th>
<th>Loc</th>
<th>MgrNo</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Administration</td>
<td>NYC</td>
<td>10011</td>
</tr>
<tr>
<td>600</td>
<td>Manufacturing</td>
<td>CHI</td>
<td>10007</td>
</tr>
<tr>
<td>500</td>
<td>Engineering</td>
<td>ATL</td>
<td>10012</td>
</tr>
<tr>
<td>300</td>
<td>Exec Office</td>
<td>NYC</td>
<td>10018</td>
</tr>
<tr>
<td>700</td>
<td>Marketing</td>
<td>NYC</td>
<td>10021</td>
</tr>
</tbody>
</table>

#### Table 11: Department Table, Personnel Database

<table>
<thead>
<tr>
<th>Project Id</th>
<th>Project Description</th>
<th>Received Date</th>
<th>Due Date</th>
<th>Compl Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>OE1-0003</td>
<td>O/E Batch System</td>
<td>82/11/21</td>
<td>83/10/27</td>
<td>83/10/27</td>
</tr>
<tr>
<td>AP2-0002</td>
<td>A/P Payable Online System</td>
<td>82/08/09</td>
<td>83/04/10</td>
<td>83/04/10</td>
</tr>
<tr>
<td>AR1-0002</td>
<td>A/R RECV Online System</td>
<td>82/08/09</td>
<td>83/04/10</td>
<td>83/04/10</td>
</tr>
<tr>
<td>OE2-0001</td>
<td>O/E Data Base Design</td>
<td>82/11/21</td>
<td>83/10/27</td>
<td>83/10/27</td>
</tr>
<tr>
<td>AR1-0003</td>
<td>A/R RECV Batch System</td>
<td>82/08/09</td>
<td>83/04/10</td>
<td>83/04/10</td>
</tr>
<tr>
<td>AR2-0001</td>
<td>A/R RECV Data Base Design</td>
<td>82/08/09</td>
<td>83/04/10</td>
<td>83/04/10</td>
</tr>
<tr>
<td>AR2-0002</td>
<td>A/R RECV Online System</td>
<td>82/08/09</td>
<td>83/04/10</td>
<td>83/04/10</td>
</tr>
<tr>
<td>AP2-0001</td>
<td>A/P Payable DB Design</td>
<td>82/08/09</td>
<td>83/04/10</td>
<td>83/04/10</td>
</tr>
<tr>
<td>AP2-0003</td>
<td>A/P Payable Batch System</td>
<td>82/08/09</td>
<td>83/04/10</td>
<td>83/04/10</td>
</tr>
<tr>
<td>PAY-0002</td>
<td>Payroll File Maintenance</td>
<td>83/01/01</td>
<td>83/12/31</td>
<td>84/01/31</td>
</tr>
<tr>
<td>OE2-0003</td>
<td>O/E Batch System</td>
<td>82/11/21</td>
<td>83/10/27</td>
<td>83/11/15</td>
</tr>
<tr>
<td>ENG-0002</td>
<td>Design Widget Pwr Supply</td>
<td>78/01/02</td>
<td>79/07/19</td>
<td>78/08/08</td>
</tr>
<tr>
<td>AP1-0001</td>
<td>A/P Payable DB Design</td>
<td>82/08/09</td>
<td>83/04/10</td>
<td>83/04/10</td>
</tr>
<tr>
<td>ENG-0003</td>
<td>Design Widget Frame</td>
<td>78/01/02</td>
<td>80/10/27</td>
<td>81/05/05</td>
</tr>
<tr>
<td>AP1-0003</td>
<td>A/P Payable Batch System</td>
<td>82/08/09</td>
<td>83/04/10</td>
<td>83/04/21</td>
</tr>
<tr>
<td>OE1-0001</td>
<td>O/E Data Base Design</td>
<td>82/11/21</td>
<td>83/10/27</td>
<td>83/10/27</td>
</tr>
<tr>
<td>OE1-0002</td>
<td>O/E Online System</td>
<td>82/11/21</td>
<td>83/10/27</td>
<td>83/10/27</td>
</tr>
<tr>
<td>AP1-0002</td>
<td>A/P Payable Online System</td>
<td>82/08/09</td>
<td>83/04/10</td>
<td>83/04/21</td>
</tr>
<tr>
<td>AR1-0001</td>
<td>A/R RECV Data Base Design</td>
<td>82/08/09</td>
<td>83/04/10</td>
<td>83/04/21</td>
</tr>
</tbody>
</table>
Preparing to Use the Sample Database

To prepare your user environment for the examples in this manual, you must first have the sample database loaded onto your client system. Then, make a separate copy of the database tables for each user.
Appendix B: Loading and Using the Sample Personnel Database

Loading the Personnel Database

Installation Script

To load the sample Personnel database on your channel-attached or network-attached client, use the PERDDL scripts. These scripts provide the BTEQ and SQL statements that install the Personnel database.

The standard location of these files depends upon the Teradata client platform you are using. For example:

- z/OS: the scripts are in the DBC.SAMPLIB or DBC.PROCLIB datasets

To load the sample Personnel database

Load the sample Personnel database on your system as follows:

1. Using your username, password, and account identifier, insert the following BTEQ LOGON command at the beginning of the PERDDL command file:

   .LOGON tdpid/DBC_username,password,acctid

2. Invoke BTEQ to execute the command file. You can enter .RUN FILE=filename, where filename identifies the input command file.

3. When you are ready to exit the session, use the LOGOFF command. Or for network-attached systems, redirect the commands through stdin to a BTEQ process, then save the results as perddl.out by entering:

   BTEQ < perddl.bteq > perddl.out

4. To verify that step 2 loaded the sample Personnel database, log on to BTEQ and enter the Teradata SQL DATABASE PERSONNEL command.

Copying the Personnel Database

Because the exercises and examples in this manual can change the data in the Personnel database tables, you should create a separate copy of the database for each user. Several users sharing the same database can invalidate the results that are shown for the examples.

Before copying the Personnel database tables, make sure that the Personnel database is installed at your site, and that you have the necessary privileges. Check with your database or system administrator if you either do not know if you have privileges or need to obtain privileges.

Note: Duplicating the examples in this manual can change the data in the sample database. If several users share a common database, their results will be neither the same, nor as presented in the examples.

Show this appendix to your database administrator (DBA) and verify that a copy of the Personnel database is available to you before you try to duplicate the examples in this manual.
Decide how you want to name the new databases you want to create. For example, you may want to name each new database according to its userid or name, that is, how the examples are presented in this manual. If you do not use this naming convention, make sure that your users know how to change the examples in the manual to access their sample Personnel database tables.

After determining the naming conventions for your sample Personnel databases, copy them as follows:

1. Use the Teradata SQL CREATE DATABASE statement to create the user’s database.
2. On channel-attached systems, after you have logged on to Teradata Database and created the user’s database, use the following command to define the ddname of the export file before logging on to BTEQ:
   - For TSO, enter:
     
     ```
     ALLOC DDNAME(ddname) DSNAME(datasetname) SHR
     ```
3. Use the following commands to export the Employee table and display its CREATE TABLE text:

   ```
   BTEQ
   .LOGON username,password
   .EXPORT FILE=[qualifier*]filename
   SHOW TABLE Personnel.Employee;
   .EXPORT RESET
   .LOGOFF
   .EXIT
   ```

   where:
   - The EXPORT command directs information from the database into a file.
   - The Teradata SQL SHOW TABLE statement displays the CREATE TABLE text for the Employee table, which is placed in the file you specified with the EXPORT command.

   The CREATE TABLE text for the Employee table is as follows:

   ```
   CREATE TABLE Employee, FALLBACK
   ( EmpNo SMALLINT FORMAT '9(5)' BETWEEN 100 AND 32001 NOT NULL,
     Name VARCHAR(12) NOT NULL,
     DeptNo SMALLINT FORMAT '999' BETWEEN 100 AND 900,
     JobTitle VARCHAR(12),
     Salary DECIMAL(8,2) FORMAT 'ZZZ,ZZ9.99' BETWEEN 1.00 AND 999000.00,
     YrsExp BYTEINT FORMAT 'Z9' BETWEEN -99 AND 99,
     DOB DATE FORMAT 'MMMbDDbYYYY' NOT NULL,
     Sex CHAR(1) UPPERCASE NOT NULL,
     Race CHAR(1) UPPERCASE,
     MStat CHAR(1) UPPERCASE,
     EdLev BYTEINT FORMAT 'Z9' BETWEEN 0 AND 22 NOT NULL,
     HCap BYTEINT FORMAT 'Z9' BETWEEN -99 AND 99 )
   UNIQUE PRIMARY INDEX (EmpNo),
   INDEX (Name); 
   ```
   - The EXPORT command with the RESET option returns control to your console.
• The EXIT command logs off Teradata Database and exits BTEQ.

4 Use your editor to insert the user’s new database name before the table name Employee on the first line of the CREATE TABLE text. For this example, the new database name is abc. In this case, the first line of text appears as:

   CREATE TABLE abc.Employee, FALLBACK

   where abc is the new database name for the Employee table.

5 Use the following commands to submit the modified export file and copy the records from the Employee table in the sample Personnel database to the abc.Employee table:

   BTEQ
   .LOGON username,password
   .RUN FILE=[qualifier]filename
   INSERT INTO userid.Employee
   SELECT * FROM Personnel.Employee;
   .LOGOFF

where:
• The BTEQ RUN command submits the modified file (which is the exported file).
• The Teradata SQL INSERT statement inserts data from the Personnel.Employee table into the abc.Employee table.
• The BTEQ LOGOFF command logs off Teradata Database session without exiting BTEQ.

6 Repeat steps 3 through 5 to copy the Charges table, using the same database name.

The CREATE TABLE text for the Charges table is as follows:

   CREATE TABLE personnel.charges,FALLBACK
   NO BEFORE JOURNAL,
   NO AFTER JOURNAL
   (
   EmpNo SMALLINT FORMAT '9(5)' TITLE 'Employee//Id' BETWEEN 10001 AND 32001 NOT NULL,
   Proj_Id CHAR(8) TITLE 'Project//Id' NOT NULL,
   WkEnd DATE TITLE 'Week//Ending',
   Hours DECIMAL(4,1)FORMAT 'ZZ9.9' BETWEEN 0.5 AND 999.5
   )
   PRIMARY INDEX (EmpNo ,Proj_Id)
   INDEX(Proj_Id);

7 Repeat steps 3 through 5 to copy the Project table, using the same database name.

The CREATE TABLE text for the Project table is as follows:

   CREATE TABLE personnel.project,FALLBACK,
   NO BEFORE JOURNAL
   NO AFTER JOURNAL
   (  
   Proj_Id CHAR(8) TITLE 'Project//Id' NOT NULL,
   Description VARCHAR(25) TITLE 'Project Description',
   RecDate DATE TITLE 'Received//Date',
   DueDate DATE TITLE 'Due//Date',
   ComDate DATE TITLE 'Compl//Date')
   UNIQUE PRIMARY INDEX(Proj_Id);

8 Repeat steps 3 through 5 to copy the Department table, using the same database name.

The CREATE TABLE text for the Department table is as follows:
Listing Salary by Location and Department

The next report lists total salary information by location and department. The examples that follow show the BTEQ input stream and equivalent commands that you can use to produce this report.
BTEQ Input Stream

Use the following BTEQ input stream to produce the report shown in Figure 4:

```
.LOGON userid,password;

DATABASE PERSONNEL;

.SET FORMAT ON
.SET WIDTH 80
.SET HEADING 'Total Salaries by Location, Department'
.SET FOOTING '&DATE &TIME||Confidential'
```

### Total Salaries by Location, Department

<table>
<thead>
<tr>
<th>Location</th>
<th>Dept. No.</th>
<th>Employee Name</th>
<th>Position</th>
<th>Salary</th>
<th>Experience</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATL</td>
<td>500</td>
<td>Carter J</td>
<td>Engineer</td>
<td>44,000.00</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Inglis C</td>
<td>Tech Writer</td>
<td>34,000.00</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Marston A</td>
<td>Secretary</td>
<td>22,000.00</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Omura H</td>
<td>Programmer</td>
<td>40,000.00</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reed C</td>
<td>Technician</td>
<td>30,000.00</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Smith T</td>
<td>Engineer</td>
<td>42,000.00</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Watson L</td>
<td>Vice Pres</td>
<td>56,000.00</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Total for Department</strong> 500</td>
<td><strong>268,000.00</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Total for Location</strong> ATL</td>
<td><strong>268,000.00</strong></td>
</tr>
<tr>
<td>NYC</td>
<td>100</td>
<td>Chin M</td>
<td>Controller</td>
<td>38,000.00</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Greene W</td>
<td>Payroll Ck</td>
<td>32,500.00</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Moffit H</td>
<td>Recruiter</td>
<td>35,000.00</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Peterson J</td>
<td>Payroll Ck</td>
<td>25,000.00</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Total for Department</strong> 100</td>
<td><strong>130,500.00</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Total for Location</strong> NYC</td>
<td><strong>386,500.00</strong></td>
</tr>
<tr>
<td>300</td>
<td></td>
<td>Leidner P</td>
<td>Secretary</td>
<td>23,000.00</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Phan A</td>
<td>Vice Pres</td>
<td>55,000.00</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Russell S</td>
<td>President</td>
<td>65,000.00</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Total for Department</strong> 300</td>
<td><strong>143,000.00</strong></td>
</tr>
<tr>
<td>700</td>
<td></td>
<td>Brangle B</td>
<td>Salesperson</td>
<td>30,000.00</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Clements D</td>
<td>Salesperson</td>
<td>38,000.00</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Smith T</td>
<td>Manager</td>
<td>45,000.00</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Total for Department</strong> 700</td>
<td><strong>113,000.00</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Total for Location</strong> NYC</td>
<td><strong>386,500.00</strong></td>
</tr>
</tbody>
</table>

02/05/28 15:57
Confidential
Appendix B: Loading and Using the Sample Personnel Database
Listing Salary by Location and Department

SELECT
   Loc               (TITLE 'Location')
   , Department.DeptNo (TITLE 'Dept./No.')
   , Name              (TITLE 'Employee//Name')
   , JobTitle          (TITLE 'Position')
   , Salary
   , YrsExp            (TITLE 'Years//Experience')
FROM
   Department
   , Employee
WHERE
   Loc IN ('NYC', 'ATL')
AND
   Salary > 15000
AND
   Department.DeptNo=Employee.DeptNo
ORDER BY
   Loc
   , Department.DeptNo
   , Name
WITH
   SUM(Salary) (TITLE 'Total for Department &2')
   , SUM(YrsExp) (TITLE ' ', FORMAT'zz9')
      BY Loc, Department.DeptNo
WITH
   SUM(Salary) (TITLE 'Total for Location &1')
   , SUM(YrsExp) (TITLE ' ', FORMAT 'zz9')
      BY Loc
WITH
   SUM(Salary) (TITLE 'GRAND TOTAL')
   , SUM(YrsExp) (TITLE ' ', FORMAT 'zz9')
;

.LOGOFF
.EXIT

where:

- LOGON logs you onto a BTEQ session.
- SET FORMAT activates BTEQ format commands.
- SET WIDTH centers the report on 80 characters (narrow printer paper).
- SET HEADING centers the specified heading on the report page.
- SET FOOTING specifies the footing on one line in three parts. The first part contains the current date, the second contains the current time, and the third is the word Confidential.
- SET SUPPRESS suppresses repeating values in column 1 (entitled Location) and column 2 (entitled Dept. No.).
- LOGOFF terminates the Teradata Database session without exiting BTEQ.
- The BTEQ substitution feature (&) inserts the column 2 value in the first subtotal line, and the column 1 value in the second subtotal line.
Using Windows or UNIX

On network-attached systems, use the following command to create the sample report shown in Figure 4 and store it in a file:

```
BTEQ <infilename>outfilename
```

where:

- `infilename` specifies the file containing the BTEQ input stream
- `outfilename` specifies the file where you want the report written.

Listing Total Earnings by Department

The report shown in Figure 5 lists total earnings information by department. The examples that follow show the BTEQ input stream and equivalent commands that you can use to produce this report.

Figure 5: Total Earnings by Department

<table>
<thead>
<tr>
<th>Dept. No.</th>
<th>Employee Name</th>
<th>Job Title</th>
<th>Salary</th>
<th>Years of Experience</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>Carter J</td>
<td>Engineer</td>
<td>$44,000.00</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Smith T</td>
<td>Engineer</td>
<td>$42,000.00</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total for Department 500</td>
<td></td>
<td>$86,000.00</td>
<td></td>
</tr>
<tr>
<td>700</td>
<td>Brangle B</td>
<td>Salesperson</td>
<td>$30,000.00</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Clements D</td>
<td>Salesperson</td>
<td>$38,000.00</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total for Department 700</td>
<td></td>
<td>$68,000.00</td>
<td></td>
</tr>
</tbody>
</table>

Note that the sample report in Figure 5 includes both a header and a footer. This example also shows you how to use a Teradata SQL macro and Teradata SQL ECHO statements. The date and the words, *Company Confidential* appear at the bottom of the page, the BTEQ SUPPRESS command is used on column 1, and the title substitution feature is used on the titles for the department subtotals.

BTEQ Input Stream

Use the following BTEQ input stream to produce the report shown in Figure 5:

```
.LOGON userid,password;
```
Appendix B: Loading and Using the Sample Personnel Database

Listing Total Earnings by Department

Basic Teradata Query Reference 341

.SET PAGELENGTH 20

CREATE MACRO mac AS (

ECHO '.SET FORMAT ON';
ECHO '.SET HEADING "Total Earnings by Department"';
ECHO '.SET FOOTING "Company Confidential||&DATE"';
ECHO '.SET SUPPRESS ON 1';

SELECT
  DeptNo   (TITLE 'Dept.//No.')
, Name     (TITLE 'Employee//Name')
, JobTitle
, Salary   (FORMAT '$$$,$9.99')
, YrsExp   (TITLE 'Years of Experience')
FROM
  Personnel.Employee
WHERE
  (JobTitle='Engineer' OR JobTitle='Salesperson')
AND
  DeptNo IN (500,700)
ORDER BY
  DeptNo
, Name
WITH
  SUM(Salary) (  FORMAT '$$$,$9.99'
                , TITLE 'Total for Department &1'
            )
  BY DeptNo
;
);

EXECUTE mac;

.LOGOFF
.EXIT

Note: The BTEQ comment statements document the contents of the input stream. They do not appear on the final report.

Using Windows or UNIX

On network-attached systems, use the following command to create the sample report shown in Figure 5 and store it in a file:

```
BTEQ <infilename>outfilename
```

where:

- `infilename` specifies the file containing the BTEQ input stream.
- `outfilename` specifies the file where you want the report written.

Using z/OS

On channel-attached systems running under z/OS, use the following z/OS JCL to produce the report shown in Figure 5:
Appendix B: Loading and Using the Sample Personnel Database

Extracting Data From Teradata Database

The following examples extract data from the Department table on Teradata Database and place it in a workstation file or a z/OS data set. The EXPORT DATA command option implies that the RECORDMODE command option is set to ON. The resulting data set/file is in normal IBM 370 format, with integers as 4-byte binary values, decimal as 370 packed decimal, and so on. (This method of unloading data can be used with multiple sessions.)

**BTEQ Input Stream**

Use the following BTEQ input stream to extract (export) data from Teradata Database:

```
LOGON userid,password;
.EXPORT DATA DDNAME=DEPTDATA*
SELECT * FROM Personnel.Department;
.EXPORT RESET
.LOGOFF
.EXIT
```

The BTEQ EXPORT command specifying the DATA option implies the RECORDMODE attribute is set to ON. The resulting dataset/file is in normal format.

**Using Windows or UNIX**

On network-attached systems, use the following command to extract data from Teradata Database and store it in a file:

```
BTEQ <infilename>outfilename
```

where:

- `infilename` specifies the file containing the BTEQ input stream.
• `outfilename` specifies the file where you want the report written.

**Using z/OS**

On channel-attached systems running under z/OS, use the following z/OS JCL to extract data from Teradata Database and store it in a file:

```
//YNEXT JOB 1,'Your Name',NOTIFY=YN,
// MSGCLASS=A,CLASS=B
//EXTRACT EXEC PGM=BTQMAIN
//STEPLIB DD DSNAME=TERADATA.APPLOAD,DISP=SHR
// DD DSNAME=TERADATA.TRLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=VBA,LRECL=137)
//SYSABEND DD SYSOUT=* 
//SAVEDATA DD DSN=YN.DEPTDATA.TEXT,DISP=(NEW,CATLG),
// UNIT=SYSDA,SPACE=(TRK,(1,1),RLSE),
// DCB=(LRECL=80,RECFM=FB,BLKSIZE=800)
//SYSTERM DD SYSOUT=* 
//SYSIN DD DATA,DLM=##
  your BTEQ script
##
```

where:

• The RECFM for SYSPRINT is VBA for z/OS.
• For SYSPRINT, the LRECL should be five greater than the number of the last character line printed, but not greater than 137. For example, to print 132 character lines, then LRECL should be 137.

**Loading Data into Teradata Database**

The following examples load a client data file onto Teradata Database. The input file used in this example is the same output file (DEPTDATA) that you produced in the preceding example.

The data load in this example uses one session. To get maximum performance, use:

• 5 sessions for each COP or network-attached PE vproc
• 10 sessions for each IFP, depending on the load placed on the AMPs

If the table has secondary indexes, you should probably use a smaller number. If the table is NO FALLBACK, you may require a larger number.

**BTEQ Input Stream**

Use the following BTEQ input stream to load (import) data into Teradata Database:

```
.LOGON 1/userid,password
.IMPORT DATA DDNAME=DEPTDATA
.REPEAT *
.USING DeptNo (SMALLINT),
    DeptName (VARCHAR(14)),
```
Loading Data into Teradata Database

```
Loc  (CHAR(3)),
MgrNo  (SMALLINT)
INSERT INTO Personnel.newtable (DeptNo, DeptName,
Loc,MgrNo)
VALUES (:DeptNo, :DeptName, :Loc, :MgrNo);
.QUIT
```

where:

- The LOGON command shows how to specify a `tdpid` other than the default (in this case, TDP1).
- The Teradata SQL USING clause describes the data in the z/OS data set.
- The REPEAT command continues the operation until the entire input file is loaded.
- The term `newtable` refers to an empty table to receive the data being loaded.

### Using Windows or UNIX

On network-attached systems, use the following command to load data from a file to Teradata Database:

```
BTEQ <infilename>outfilename
```

where:

- `infilename` specifies the file containing the BTEQ input stream
- `outfilename` specifies the file for saving the results of running the input stream commands specified by `outfilename`.

### Using z/OS

On channel-attached systems running under z/OS, use the following z/OS JCL to load data from a client file into Teradata Database:

```
//YNMULTI JOB1,'Your Name',NOTIFY=YN,
// MSGCLASS=A,CLASS=B
//BULK EXEC PGM=BTQMAIN,REGION=2048K
//STEPLIB DD DSNAME=TERADATA.APPLOAD,DISP=SHR
// DD DSNAME=TERADATA.TRLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=VBA,LRECL=137)
//SYSPRINT DD SYSOUT=*
//DEPTDATA DD DSN=YN.DEPTDATA.TEXT,DISP=SHR
//SYSIN DD DATA,DLM=##
//SYSTERM DD SYSOUT=*#
```

where:

- The RECFM for SYSPRINT is VBA for z/OS.
- For SYSPRINT, the LRECL should be five greater than the number of the last character line printed, but not greater than 137. For example, to print 132 character lines, then LRECL should be 137.
Using a z/OS Batch Job to Extract and Load Data

The following z/OS JCL example selects data from the Employee table and saves it in a client data set (SAVEDATA). The example selects rows according to values of EmpNo contained in an input data set (DATA). (EmpNo is the prime key for the Employee table.)

```
//YNINPUT JOB1,'Your Name',CLASS=B,NOTIFY=YN,
  //               MSGCLASS=A,MSGLEVEL=(1,1)
//BTEQ EXECPGM=BTQMAIN
//STEPLIB DD DSN=TERADATA.APPLOAD,DISP=SHR
//DD DSN=TERADATA.TRLOAD,DISP=SHR
//INFILE DD DSN=YN.BTEQ.CNTL(DATA),DISP=SHR
//SYSPRINT DD SYSOUT=*   //SYSSABEND DD SYSOUT=*   //SYSTEM DD SYSOUT=*   //SAVEDATA DD DSN=YN.SAVEDATA.TEXT,DISP=(NEW,CATLG),   //   UNIT=SYSDA,SPACE=(TRK,(1,1),RLSE),   //   DCB=(LRECL=80,RECFM=FB,BLKSIZ=800)
//SYSIN DD DATA,DLM=##
          .IMPORT DATA DDNAME=INFILE
          .EXPORT DATA DDNAME=SAVEDATA
          .SET SESSIONS 2
          .LOGON userid,password
          .REPEAT 10
          USING EMPNO  (CHAR(5)),
            FILLER (CHAR(75))
          SELECT * FROM PERSONNEL.EMPLOYEE
          WHERE EMPNO = :EMPNO;
          .EXPORT RESET
          .QUIT
##
```

where:

- The Teradata SQL script consists of a single SELECT statement preceded by a USING modifier. The modifier describes the data that is used to qualify rows selected from Employee. (Note that the USING modifier must precede the Teradata SQL statement that it modifies.)
- The IMPORT command reads data from the DATA data set.
- The EXPORT command sends the data selected to the SAVEDATA data set.
- The SESSIONS command opens two sessions to process the job.
- The LOGON command logs the user on to the number of Teradata Database sessions specified by the SESSIONS command.
- The REPEAT command repeats the SELECT request 10 times.
- The EXPORT RESET statement cancels the export function after the select operation. Any processing results are now directed to SYSPRINT. (In this example, this command is optional because the next command is QUIT.)
- The QUIT command logs you out of BTEQ and returns you to the system prompt.
Listing the Error Log File

The figure below shows a sample Teradata Database error log table.

Figure 6: Error Log Listing

<table>
<thead>
<tr>
<th>ErrDate</th>
<th>ErrTime</th>
<th>Processor</th>
<th>Event</th>
<th>LineNumber</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>82/01/01</td>
<td>00:00:01.950</td>
<td>002:6 2497</td>
<td>1</td>
<td>On 82/01/01 00:00</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00:00:02.030</td>
<td>002:6 2497</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00:00:02.040</td>
<td>002:6 2492</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*** Warning: RetLimit exceeded.***

*** Ignoring the rest of the output for this statement.***

BTEQ -- Enter your SQL request logon or BTEQ command:

.LOGOFF

*** You are now logged off from the DBC.***

BTEQ Input Stream

Use the following BTEQ input stream to produce the report shown in Figure 6:

```sql
.SET WIDTH 132
.SET RTITLE 'The Error log by date and time'
.SET NULL AS ' '
.SET SUPPRESS ON 1,2,3,4
.SET UNDERLINE ON 1,2,3,4
.SET SEPARATOR ' | '
.SET RETLIMIT 20

SELECT ErrDate, ErrTime, Processor(title 'Pro//ces//sor'),
       Event(title 'Evnt'), LineNumber(title 'Line//Num-//ber'),
       Text(format 'X(78)')
FROM DBC.Errorlog
WHERE LineNumber <> 0
ORDER BY ErrDate,ErrTime,Processor,Event,LineNumber;
.LOGOFF
.EXIT
```

where:
• The UNDERLINE command sets each entry apart in the error log, since each entry consists of several rows.
• The report is designed for 132-column paper.
• For SYSPRINT, the LRECL should be five greater than the number of the last character line printed, but not greater than 137. For example, to print 132 character lines, then LRECL should be 137.

Using Windows or UNIX

On network-attached systems, use the following command to produce the error log shown in Figure 6 and store it in a file:

```bash
BTEQ <infilename> outfilename
```

where:

• `infilename` specifies the file containing the BTEQ input stream.
• `outfilename` specifies the file for storing the error log.

Using z/OS

On channel-attached systems running under z/OS, use the following z/OS JCL to produce the error log shown in Figure 6 and store it in a client file:

```zos
//YNERR JOB1,'Your Name',NOTIFY=YN,MSGCLASS=A,CLASS=B
//BTEQ EXECPGM=BTQMAIN
//STEPLIB DDDSN=TERADATA.APPLOAD,DISP=SHR
// DDDSNNAME=TERADATA.TRLOAD,DISP=SHR
//SYSPRINT DDSYSOUT=*,DCB=(RECFM=VBA,LRECL=137)
//SYSEABEND DDSYSOUT=* 
//SYSTEM DDSYSOUT=* 
//SYSTIN DDDATA,DLM=##
  your BTEQ script
##
```
Appendix B: Loading and Using the Sample Personnel Database
Listing the Error Log File
This appendix describes differences between channel-attached and network-attached systems.

Commands and Keywords

Sometimes, BTEQ for channel-attached systems uses different commands or keywords than BTEQ for network-attached systems, or the same command functions differently. The next table lists, in alphabetical order, equivalent commands and keywords for each type of system:

<table>
<thead>
<tr>
<th>Commands or Keywords: BTEQ for Channel-Attached Systems</th>
<th>Commands or Keywords: BTEQ for Network-Attached Systems</th>
<th>Similarities Between BTEQ for Channel-Attached Systems and BTEQ for Network-Attached Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>The = command does not prompt you for the default n.</td>
<td>The = command prompts you for the default n.</td>
<td>On both channel-attached and network-attached systems, the default for n is 1 if you do not provide a value for n.</td>
</tr>
<tr>
<td>TSO</td>
<td>OS</td>
<td></td>
</tr>
<tr>
<td>DDNAME</td>
<td>FILE</td>
<td>FILE and DDNAME can be used interchangeably on both channel-attached and network-attached systems.</td>
</tr>
<tr>
<td>The EXPORT command underlines the summary values from WITH clauses on DIF files.</td>
<td>The EXPORT command does not underline the summary values from WITH clauses on DIF files.</td>
<td></td>
</tr>
<tr>
<td>Setting the FORMAT command option to ON changes the width in batch mode.</td>
<td>Setting the FORMAT command option to ON does not change the width.</td>
<td></td>
</tr>
<tr>
<td>The HANG command pauses batch jobs on z/OS.</td>
<td>The HANG command pauses both interactive and batch jobs.</td>
<td>The HANG command specifies time in seconds on both channel-attached and network-attached systems.</td>
</tr>
</tbody>
</table>
### Appendix C: Differences Between Channel-Attached and Network-Attached Systems

#### tdpid Differences

<table>
<thead>
<tr>
<th>Commands or Keywords: BTEQ for Channel-Attached Systems</th>
<th>Commands or Keywords: BTEQ for Network-Attached Systems</th>
<th>Similarities Between BTEQ for Channel-Attached Systems and BTEQ for Network-Attached Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>The IF...THEN... command can contain either a BTEQ command or a Teradata SQL request in the THEN clause.</td>
<td>In a Teradata SQL macro, the IF...THEN... command can only contain a BTEQ command in the THEN clause.</td>
<td>The BTEQ invocation for network-attached systems may use several keywords (&lt;, &gt;, &gt;&gt;, &gt;&amp;1, &gt;&amp;).</td>
</tr>
<tr>
<td>A channel-attached system keyword does not exactly correspond to them.</td>
<td>The BTEQ invocation for network-attached systems may use several keywords (&lt;, &gt;, &gt;&gt;, &gt;&amp;1, &gt;&amp;).</td>
<td></td>
</tr>
<tr>
<td>IMPORT command does not have a REPORT keyword.</td>
<td>IMPORT command has a REPORT keyword.</td>
<td></td>
</tr>
<tr>
<td>In the NOTIFY command, the QUEUE option is only supported for z/OS clients.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The PAGEBREAK command clears the screen on intelligent terminals.</td>
<td>The PAGEBREAK command does not clear the screen on ASCII terminals.</td>
<td></td>
</tr>
<tr>
<td>The REPEAT command terminates BTEQ under NOTERM or z/OS.</td>
<td></td>
<td>On channel-attached and network-attached systems using TSO, the REPEAT command:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Does not resubmit a “kill request”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Provides an explanation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Aborts anything taking place in the BTEQ session</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Leaves the user in BTEQ</td>
</tr>
</tbody>
</table>

#### tdpid Differences

There are also differences regarding the *tdpid* between BTEQ for channel-attached systems and BTEQ for network-attached systems:

<table>
<thead>
<tr>
<th>tdpid: BTEQ for Channel-Attached Systems</th>
<th>tdpid: BTEQ for Network-Attached Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>The <em>tdpid</em> is the identifier of the TDP that handles Teradata Database traffic.</td>
<td>The <em>tdpid</em> is the identifier (the network ID) of Teradata Database.</td>
</tr>
<tr>
<td>You can abbreviate the <em>tdpid</em> to one character if your site is using the four-character TDP naming convention.</td>
<td>You cannot abbreviate the <em>tdpid</em> to one character.</td>
</tr>
</tbody>
</table>
This appendix describes the Kanji, Chinese, Korean, and Unicode character set use on channel-attached or network-attached clients.

Kanji Character Sets for Channel-Attached Systems

BTEQ on the mainframe accepts data specified in the following character set names:

- KANJIEBCDIC5026_0I
- KANJIEBCDIC5035_0I
- KATAKANAEBCDIC
- EBCDIC

Supporting a particular character set allows you to specify identifiers and data in that character set.

Character set names must be specified in uppercase.

Kanji Character Sets for Network-Attached Systems

Kanji support is an optional feature of BTEQ on a Kanji Windows system (refers to a PC with a Kanji keyboard, Kanji Windows for DOS, and Kanji language support software), which uses only the KanjiShift-JIS character set.

Establishing the Character Set

To establish a character set, you can do any of the following:

- Establish a character set that is in effect for a BTEQ job.
- Override the default character set.
- Switch between multi-byte and English modes.
Appendix D: Kanji, Chinese, Korean, and Unicode Character Set Support
Kanji Character Sets for Network-Attached Systems

The character set is preserved for the duration of the session (until a QUIT), not just for the logon.

You can establish a character set in one of the following ways:

- Specify the character set with the BTEQ SESSION CHARSET command.
  You do not need to issue the command when you log on to BTEQ. Issue the command before you run the operation that needs to be in a particular character set. This user-specified character set will override all other character sets previously specified or placed in default.
  For example:

  .SET SESSION CHARSET 'KANJIEBCDIC5035_0I'

  At logon, for example, if the logon string is normally in the form:

  BTEQ .LOGON
  tdpid
  /user,password

  you could modify it to use KanjiEBCDIC as follows:

  BTEQ .SET SESSION CHARSET 'EBCDIC';
  .LOGON tdpid/user,password

  **Note:** The .SET string must precede the .LOGON string.

- Specify a character set value in:
  - The HSHSPB parameter module for IBM mainframes
  - The clispb.dat file for UNIX platforms
    This takes second precedence.
  - If a character set has not been user-specified or specified in HSHSPB or clispb.dat, then the default is the value in the system table, DBC.Hosts.

If an application relies on the DBC.Hosts table for the default character set name, you must ensure that the initial logon from the channel-attached side is in KanjiEBCDIC; otherwise, the default character set cannot be known to the application before logon.

**Description**

To correctly display Katakana and double-byte data on z/OS, enter data using PDF/EDIT, and view the output with PDF/BROWSE

BTEQ has no special handling for the extra spaces required for the SO/SI characters (that is, Shift-Out/Shift-In).

Be aware that lowercase English letters are not allowed when in KATAKANAEBCDIC mode.

If your BTEQ job contains lowercase English characters, it can be executed using the one of the character sets that supports lowercase Latin letters, such as any of the following character sets:

- KANJIECDIC5026_0I
- KANJIECDIC5035_0I
- EBCDIC
Note: If the job contains only uppercase English characters, it can be executed using any character set.

Message Display

BTEQ generates all messages in uppercase Latin characters when a character set that does not support lowercase Latin letters is the current character set. Do not alter message insertions from the input script or from Teradata Database.

SESSION CHARSET Command

The SESSION CHARSET command specifies which character set is in effect during a specific BTEQ invocation. KANJIEUC_0U and KANJISJIS_0S are two of many possible choices of the SESSION CHARSET command. If the character set is not specified, the default is ASCII.

Syntax

```
SET SESSION CHARSET 
ASCII 
_KANJIEUC_0U 
_KANJISJIS_0S 
n
```

where:
- ASCII is the default character set.
- KANJIEUC_0U specifies the Kanjii Extended UNIX Code character set.
- KANJISJIS_0S specifies the combined JIS-x208 and JIS-x0201 character set developed by Microsoft.
- \( n \) is a character set code in the range of 0 through 255. (118, for example, is the character set code for KANJIEUC_0U, and 119 is the character set code for KANJISJIS_0S.)

GRAPHIC, VARGRAPHIC and LONG VARGRAPHIC

GRAPHIC data types are used to define multi-byte character set data. BTEQ accepts GRAPHIC data in its input data file. Two modified Teradata SQL statements support GRAPHIC, VARGRAPHIC, and LONG VARGRAPHIC input data:

- DEFINE
- SHOW
Chinese and Korean character sets are supported for mainframe and network platforms.

### Chinese Character Sets - Teradata Defined

<table>
<thead>
<tr>
<th>Chinese Character Sets - Teradata Defined</th>
<th>Number Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEBCDIC935_21J</td>
<td>109</td>
</tr>
<tr>
<td>TCHEBCDIC937_3IB</td>
<td>110</td>
</tr>
<tr>
<td>SCHGB2312_1T0</td>
<td>121</td>
</tr>
<tr>
<td>TCHBIG5_1R0</td>
<td>122</td>
</tr>
</tbody>
</table>

### Korean Character Sets - Teradata Defined

<table>
<thead>
<tr>
<th>Korean Character Sets - Teradata Defined</th>
<th>Number Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>HANGULEBCDIC933_1II</td>
<td>108</td>
</tr>
<tr>
<td>HANGULKSC5601_2R4</td>
<td>120</td>
</tr>
</tbody>
</table>

If the predefined character sets are not appropriate for your site, you can define the character sets in the following tables (refer to *SQL Fundamentals* and *International Character Set Support* for information on defining your own character set).

### Chinese Character Sets - Site Defined

<table>
<thead>
<tr>
<th>Chinese Character Sets - Site Defined</th>
<th>Number Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDSCHEBCDIC935_6IJ</td>
<td>75</td>
</tr>
<tr>
<td>SDTCHEBCDIC937_7IB</td>
<td>76</td>
</tr>
<tr>
<td>SDTCHGB2312_2T0</td>
<td>94</td>
</tr>
<tr>
<td>SDTCHBIG5_3R0</td>
<td>95</td>
</tr>
</tbody>
</table>

### Korean Character Sets - Site Defined

<table>
<thead>
<tr>
<th>Korean Character Sets - Site Defined</th>
<th>Number Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDHANGULEBCDIC933_5II</td>
<td>74</td>
</tr>
<tr>
<td>SDHANGULKSC5601_4R4</td>
<td>93</td>
</tr>
</tbody>
</table>

### Rules

The following rules exist for Chinese and Korean character sets on both network and mainframe platforms. For information on Chinese and Korean character set restrictions for Teradata Database, refer to the Teradata Database documentation.
SHOW TABLE Command

When you use the SHOW TABLE command, compress and default values that include any characters outside of 7-bit ASCII are shown as Unicode hexadecimal constants. Also, the SHOW TABLE command cannot show a table that has object names with characters outside of 7-bit ASCII.

Maximum String Length

The DBS requires two bytes to process each of the Chinese or Korean characters. This limits both request size and record size. For example, if a record consists of one string, the length of that string is limited to a maximum of 32,000 characters, or 64,000 bytes.

Unicode Character Set

The following UTF8 and UTF16 character sets are available:

<table>
<thead>
<tr>
<th>Unicode Character Sets</th>
<th>Number Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTF8</td>
<td>63</td>
</tr>
<tr>
<td>UTF16</td>
<td>62</td>
</tr>
</tbody>
</table>

Rules

It is important to understand that a Unicode character can be from one to four bytes in length, depending upon if the session character set is UTF8 or UTF16. Therefore, the size of output/export files is not indicative of the number of characters it contains.

It is the user’s responsibility to ensure that the endianness of any UTF16 input files are the same as the endianness of the platform BTEQ is running on. If not, or if an incorrect BOM is encountered, BTEQ will report an error.

Filenames

All filenames (COMPILE, EXPORT, IMPORT, MESSAGEOUT, or RUN) must be defined using only basic LATIN characters.

Network-Attached Systems

To start a UTF8 or UTF16 session, it is recommended that the -c option be used to define the session charset encoding, and possibly the -e option (batch mode) or -m option (interactive mode) to define the I/O encoding.

A BOM is optional for the following input files:

- A file redirected through stdin
- RUN file
- IMPORT REPORT files
- IMPORT VARTEXT files
Stored Procedure files
A BOM is optional for the following output files:
- MESSAGEOUT file
- EXPORT REPORT files
- EXPORT DIF files

BTEQ does not allow for a BOM to be written to stdout or stderr.

**Channel-Attached Systems**

z/OS BTEQ supports Unicode sessions in the following way:
- Input (SYSIN and RUN files) will be read as EBCDIC
- Output (SYSOUT and MESSAGEOUT files) will be written in EBCDIC
- IMPORT VARTEXT files must be in the session character set encoding (UTF8 or UTF16). A BOM is optional.
- EXPORT REPORT/DIF files will be written in the session character set encoding (UTF8 or UTF16)

**Caution:** The EBCDIC repertoire is much smaller than Unicode. Trying to display Unicode characters not in the EBCDIC repertoire to SYSOUT (or a MESSAGEOUT file) will result in a translation error.
This appendix describes how to use BTEQ for Windows, called BTEQWIN.

**What is BTEQWIN?**

BTEQWIN is a Windows version of standard Teradata BTEQ that incorporates a standard Windows interface.

BTEQWIN allows you to:

- Scroll back and forth through BTEQ output.
- Save all BTEQ output to a file of your choice.
- Modify and resubmit SQL statements to Teradata Database without having to retype them.
- Initiate an ASCII, UTF8, UTF16, or KANJISJIS_OS session charset.
- Properly input and view multi-byte characters
- Use various fonts

**BTEQWIN Main Window**

When BTEQWIN is first started, it prompts you for the session charset to use, as shown in the following figure.
The BTEQWIN user interface consists of a standard application window, as shown in Figure 8.
The following functions can be performed from the BTEQWIN Main Window:

<table>
<thead>
<tr>
<th>You can…</th>
<th>For…</th>
</tr>
</thead>
<tbody>
<tr>
<td>enter commands</td>
<td>session control, result formatting, and output data handling.</td>
</tr>
<tr>
<td>submit Teradata SQL statements</td>
<td>processing.</td>
</tr>
<tr>
<td>and requests to the associated</td>
<td>Statements can be specified interactively, a statement at a time, or in batch mode using macros.</td>
</tr>
<tr>
<td>Teradata Database</td>
<td></td>
</tr>
</tbody>
</table>

All input and output is displayed in the window. The command line, identified by a blinking cursor, is always the line that follows the last command prompt.

**BTEQWIN Menus**

**BTEQWIN Menu Bar**

The BTEQWIN menu bar contains the following menus:

- Sessions
• Edit
• View
• Window
• Help

**Sessions Menu**
The commands on the initial Sessions menu are for session control, printing, and setting up printing parameters. The menu also includes a command to exit BTEQWIN.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>New</td>
<td>Starts a new BTEQWIN session.</td>
</tr>
<tr>
<td>Interrupt</td>
<td>Interrupts a BTEQ task.</td>
</tr>
<tr>
<td>Abort</td>
<td>Aborts the current active BTEQWIN session before the routine termination.</td>
</tr>
<tr>
<td>Close</td>
<td>Closes the current BTEQWIN session.</td>
</tr>
<tr>
<td>Save As</td>
<td>Saves the currently displayed information to a file.</td>
</tr>
<tr>
<td>Print</td>
<td>Prints the contents of the active window.</td>
</tr>
<tr>
<td>Print Preview</td>
<td>Displays a preview of the information selected for printing.</td>
</tr>
<tr>
<td>Print Setup</td>
<td>Allows you to reconfigure your current printer setup.</td>
</tr>
<tr>
<td>Exit</td>
<td>Exits BTEQWIN.</td>
</tr>
</tbody>
</table>

**Edit Menu**
The Edit menu allows you to:

• Cut, copy and paste text in BTEQWIN windows.
• Edit or send your selected text again.
• Select or deselect all text shown.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy</td>
<td>Allows you to copy the selected text into an internal buffer. This option is not available if there is no text selected.</td>
</tr>
<tr>
<td>Paste</td>
<td>Allows you to paste the contents of the internal buffer onto the window where the cursor is positioned. This item is not available if the internal buffer is empty.</td>
</tr>
</tbody>
</table>
Appendix E: Using BTEQWIN

BTEQWIN Menus

View Menu
The View menu allows you to:

- Enable or disable the toolbar and status bar.
- Select the display font.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear All</td>
<td>Allows you to clear the entire contents of the currently active window.</td>
</tr>
<tr>
<td>Edit Selection</td>
<td>Displays the SQL Editor window.</td>
</tr>
<tr>
<td></td>
<td>(You can also press F2 or CTRL+E to display the SQL Editor window.)</td>
</tr>
<tr>
<td></td>
<td>• If text was previously selected in the BTEQ window, the text is displayed in the SQL Editor window.</td>
</tr>
<tr>
<td></td>
<td>• If text was not selected, the window is empty.</td>
</tr>
<tr>
<td></td>
<td>After editing, you can press Send Selection Again to submit the statements to BTEQ.</td>
</tr>
<tr>
<td>Send Selection Again</td>
<td>Submits any highlighted text (such as an SQL statement or BTEQ command) to BTEQ.</td>
</tr>
<tr>
<td></td>
<td>If text is not selected, nothing is submitted.</td>
</tr>
<tr>
<td></td>
<td>(You can also press F3 or CTRL+A to submit the highlighted text.)</td>
</tr>
<tr>
<td>Select All</td>
<td>Selects the entire contents of the currently active window.</td>
</tr>
</tbody>
</table>

Window Menu
The Window menu allows you to manage your BTEQWIN windows.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cascade</td>
<td>Overlap the windows starting in the upper left corner.</td>
</tr>
<tr>
<td>Tile</td>
<td>Display all windows so that they are all visible.</td>
</tr>
<tr>
<td>Arrange Icons</td>
<td>Reorganize the minimized window icons.</td>
</tr>
</tbody>
</table>

Help Menu
The Help menu provides useful information to the user.
Before BTEQWIN can display multi-byte Unicode and Kanji characters on the screen, you need to reset the display font.

1. From the BTEQWIN menu, select View -> Font.
2. Choose one of the following fonts
   - Arial Unicode MS
   - Courier New
   - Lucinda Sans Unicode
   - Microsoft Sans Serif
   - Tahoma

### Displaying Multi-Byte Characters

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Help Topics</td>
<td>Displays online documentation for BTEQWIN.</td>
</tr>
<tr>
<td>About BTEQWIN</td>
<td>Displays release information for BTEQWIN.</td>
</tr>
</tbody>
</table>
A

abend  Abnormal END of task. Termination of a task prior to its completion because of an error condition that cannot be resolved by the recovery facilities that operate during execution.


B

BOM  Byte Order Mark. A two or three byte field at the beginning of a Unicode file indicating the byte order of the text. A BOM is optional for Unicode text files. A BOM is not allowed in files containing both Unicode text and binary data. A UTF-8 BOM = 0xEFBBBF. A UTF-16LE BOM = 0xFFFE. A UTF-16BE BOM = 0xFEFF.

BTEQ  Basic Teradata Query facility. A utility that allows users to access data on Teradata Database, and format reports for both print and screen output.

C

channel-attached  A mainframe computer that communicates with a server (for example, Teradata Database) through a channel driver.

CLI  Call-Level Interface. The interface between the application program and the MTDP (for network-attached clients) or TDP (for channel-attached clients).

CLIv2  Call-Level Interface Version2

client  A computer that can access Teradata Database.

client-server environment  The distribution of work on a LAN in which the processing of an application is divided between a front-end client and a back-end server, resulting in faster, more efficient processing.

D

DIF  Data Interchange Format. An export format supported by BTEQ, DIF is a text format used by many spreadsheet applications as a program-independent method for storing row and column data.

DBA  Database Administrator

DBC  Database Computer

DOS  Disk Operating System
F

Field Mode  A Teradata Database response mode which returns database fields (except for large objects) formatted as character data.

FIPS  Federal Information Processing Standards

FastExport  Fast Data Export utility. A program that transfers large amounts of data from Teradata Database to a network-attached or channel-attached client.

FastLoad  Fast Data Load utility. A program that loads empty tables on Teradata Database with data from a network-attached or channel-attached client.

G

gateway  A device that connects networks having different protocols.

I

Indicator Mode  A Teradata Database response mode which returns all fields (except for large objects) as unformatted data.

I/O  Input/Output

J

JCL  Job Control Language

L

LAN  Local Area Network

M

macro  a file that is created and stored on Teradata Database, and is executed in response to a Teradata SQL EXECUTE statement

Multipart Indicator Mode  A Teradata Database response mode which returns all fields (including large objects) as unformatted data.

MTDP  Micro Teradata Director Program

MultiLoad  Multiple Table Update utility. A program that executes specified insert, update, and delete operations against multiple Teradata Database tables.

N

network  In the context of Teradata Database, a LAN (see LAN).

network attached  A computer that communicates over the LAN with a server (for example, Teradata Database).
**P**

**protocol**  The rules for the format, sequence and relative timing of messages exchanged on a network.

**PDE**  Parallel Database Extension

**R**

**Record Mode**  A Teradata Database response mode which returns non-null fields (except for large objects) as unformatted data.

**records**  A formatted record, in Teradata Database terms, consists of a record created by a Teradata Database utility, such as BTEQ, where the record is packaged with begin- and end-record bytes specific to Teradata Database. Unformatted records are any records not originating on Teradata Database. These files contain records that must be defined before loading onto Teradata Database.

**row**  Used to represent a record in a file or a tuple in a relation.

**run file**  A script that is not contained within the SYSIN file, but rather executed through use of the .RUN BTEQ command.

**S**

**script**  A file that contains a set of BTEQ commands and/or SQL statements.

**session**  A session begins when the user logs on to Teradata Database and ends when the user logs off Teradata Database. Also called a Teradata Database session.

**SQL**  (Structured Query Language) See Teradata SQL.

**stored procedure**  A file that is created, compiled and stored on Teradata Database, and is executed in response to a Teradata SQL CALL statement.

**T**

**TDP**  Teradata Director Program

**TDPID**  Teradata Director Program Identification

**TSO**  Time Sharing Option

**Teradata SQL**  The Teradata Database dialect of the relational language SQL, having data definition and data manipulation statements. A data definition statement would be a CREATE TABLE statement and a data manipulation statement would be a data retrieval statement (a SELECT statement).

**U**

**unformatted records**  See Records.
unicode  The universal character encoding, maintained by the Unicode Consortium
http://www.unicode.org. The encoding standard provides the basis for processing, storage,
and interchange of text data in any language in all modern software and information
technology protocols. Within this manual, Unicode refers to either UTF8 or UTF16.

UTF8   Code scheme that uses a sequence of one to four bytes to represent each Unicode
scalar value.

UTF16  Code scheme that uses a single 16-bit code unit to represent the most common 63k
Unicode scalar values, and a pair of 16-bit code units, called surrogates, to encode
supplementary characters in Unicode.

V

varbyte A data type that represents a variable-length binary string.

varchar A data type that represents a variable-length non-numeric character.

vargraphic  A data type that represents a variable-length string of characters.

W

workstation A network-attached client.

Z

z/OS  Z Series Operating System. An IBM mainframe operating system. One of the primary
operating systems for large IBM computers.
Symbols
* (asterisk), comment character 50
<cr> (carriage return), comment character 50
<lf> (line feed), comment character 50
= command 126
and REPEAT command 253
using interactively 79

A
abends 53, 90, 92, 93, 213, 218
I/O errors 92
ABORT command 81, 83, 130
access modules. See AXSMOD
acctid 40
ACTIVITYCO keyword
EXIT command 151
QUIT command 245
ACTIVITYCOUNT keyword
IF...THEN... command 189
ANSI mode
comments 50
COMPILE command 136
REPEAT command 255
SESSION TRANSACTION command 284
stored procedures in 75
AUTOKEYRETRIEVE 131
automatic file allocation 39
AXSMOD
examples 196
keyword for EXPORT command 153
keyword for IMPORT command 191

B
batch mode 24, 52
defining input & output files 54
LOGON command 206
specifying standard output file 56
switching from interactive 54
begin sessions 28
BEGIN TRANSACTION statement 252
blank lines
inserting in reports 103
suppression of 164
BLKSIZE
specifying for output file 57

used in exports 160
block size. See BLKSZE
BOM
   COMPILE command 135
   EXPORT command 153
   IMPORT command 193
   keyword in MESSAGEOUT command 214
   RUN command 272
   Unicode 135, 153, 193, 272
branching and testing commands 75
break function 81, 82
BTET, SESSION TRANSACTION command 284

C
Call-Level Interface (CLI) 27
channel-attached systems
caller set support 351
diagram 25
differences from network-attached systems 349, 350
operating system command 84
character set support
description 351
rules 354
SESSION CHARSET command 353
Chinese character set support 351, 354
client-server environment 24
CLOSE keyword, EXPORT command 156
columns
   folding 163
   maximum number of 262
columns
   command
   continuing 123
echoed 66
execution order 62
file control 30
for controlling format 32
for controlling sequence 31
for controlling sessions 28
format 121
list of BTEQ 28
sequence, changing in reports 115
syntax 121
terminating 124
command line options 37, 276
commands
   = 126
ABORT 130
COMPILE 134
DEFAULTS 140
ECHOREQ 145
ERRORLEVEL 147
ERROROUT 149
EXIT 151
EXPORT 152
FOLDLINE 163
FOOTING 107, 169
FORMAT 174
FORMCHAR 176
FULLYEAR 178
GOTO 179
HALT EXECUTION (HX) 187
HANG 180
HEADING 107, 181
HX 187
IF...THEN... 188
IMPORT 190
INDICDATA 197
LABEL 200
LOGOFF 205
MAXERROR 213
MESSAGEOUT 214
NOTIFY 216
NULL 229
OMIT 232
OS 235
PAGEBREAK 238
PAGELENGTH 241
QUIET 243
QUIT 245
RECORDMODE 246
REMARK 249
REPEAT 252
REPEATSTOP 257
REPORTALIGN 258
RETCANCEL 260
RETLIMIT 262
RETRY 265
RTITLE 107, 267
RUN 271
SEPARATOR 273
SESSION SQLFLAG 282
SESSION TRANSACTION 284
SESSIONS 287
SET FOLDLINE 163
SET FOOTING 169
SET FORMCHAR 176
SET HEADING 181
SET NULL 229
SET PAGEBREAK 238
SET PAGELength 241
SET QUIET 243
SET RECORDMODE 246
SET RETCANCEl 260
SET RETLIMIT 262
SET RETRY 265
SET RTITLE 267
SET SEPARATOR 273
SET SESSION CHARSET 276
SET SESSION RESPBUFLEN 280
SET SESSIONS 287
SET SIDETTILES 299
SET SKIPDOUBLE 302
SET SKIPLINE 304
SET SUPPRESS 307
SET TDP 310
SET TIMEMSG 312
SET TITLEDASHES 315
SET UNDERLINE 319
SET WIDTH 322
SHOW CONTROLS 289
SHOW ERRORMAP 295
SHOW VERSIONS 297
SIDETTILES 299
SUPPRESS 102, 307
TDP 310
TITLEDASHES 312, 315
TSO 318
types 28
UNDERLINE 319
WIDTH 322
comments 50
compatibility, cross-platform 136
compilation errors, stored procedures 85
COMPILE command 134
conditional message 78
configuration
channel-attached systems 25
network-attached systems 26
constant character, EXPORT command 158
CONTINUE keyword 130
continuing BTEQ commands 123
control settings, LOGOFF 205
controlling
command sequences 31
format 32
input files 30
output files 31
sessions 28
converting displays to reports 115
CREATE PROCEDURE privilege 135
creating
BTEQ script 62
run file 63

D
DATA keyword
   EXPORT command 153
   IMPORT command 191
data loss condition 85
data types
   GRAPHIC 353
   IMPORT command 194
   INDICDATA command 197
   LONG VARGRAPHIC 353
   multi-byte character set data 353
   numeric or decimal overflow 99, 247
   RECORDMODE command 246
   VARGRAPHIC 353
DatalInfo parcel 197
DATALABELS keyword, EXPORT command 154
date, using RTITLE 115
ddname 54, 59, 60, 64, 65, 71, 92, 123, 149, 154, 191
defining in export file 71
DECIMALDIGITS 138
default
   formatting options 95
   specifications 95
DEFAULT keyword, FORMCHAR command 176
DEFAULTS command 140
deleting duplicated data 102
DIF format 117
DIF keyword, EXPORT command 154
displays, converting to reports 115
DOS
   operating system command from BTEQ 84
double-break function 81
duplicated data, deleting 102

E
EBCDIC characters 119
echo required function, disabling 96
echoing commands 66
ECHOREQ command 96, 145
editor
   for scripts and run files 62
   system 65
EJECT 154
end sessions 28
ENTRY keyword, SESSION SQLFLAG 282
ERRORCODE keyword
   EXIT command 151
   IF...THEN... command 189
   QUIT command 245
ERRORLEVEL command 90, 147
ERRORLEVEL keyword
EXIT command 151
IF...THEN... command 189
QUIT command 245
ERROROUT command 149
errors
   error codes 85
   exiting options 43
   mapping to severity levels 295
   maximum level 85, 90
   output file 59, 91
   severity level 85, 90, 147, 213
   stored procedures 91
executing a run file 64
EXIT command 43, 151
EXIT keyword, NOTIFY command 216
exiting BTEQ 151
EXPORT command
   described 152
   resetting 116

F
FastLoad format output file 153
Field Mode
   controlling formats 33
   controlling output files 31
default 98
   EXPORT command 159
IMPORT command 192, 194
in export file 118
in REPORT format 118
INDICDATA command 197, 198, 199
numeric overflow 99
RECORDMODE command 246
response mode 91
returning to 99
setting 101
   with echo 66
file allocation, automatic 39
file control commands 30
filename, quotes with 121
files, creating and modifying 61
FOLDLINE command 163
footers in reports 107
FOOTING command 107, 169
form feed in reports 110
format
   BTEQ output 95
   control commands 32
dashed lines 319
   folds and breaks 164
   page breaks 104
   page size 109
   phrase 158
record, with IMPORT 192
report 302, 304
title at page top 115
FORMAT command 174
FORMAT phrase, EXPORT command 158
FORMCHAR command 176
FORMCHAR keyword, DEFAULTS command 140
FULLYEAR command 178

G
GOTO command 179
GRAPHIC data types 353

H
HALT EXECUTION (HX) command 187
HANG command
described 180
headers in reports 107
HEADING command 107, 181
help for using BTEQ 46
HIGH keyword, NOTIFY command 216
horizontal display of data 102
host/workstation differences 349
HX command 187

I
I/O errors 90, 92, 213
IF...THEN command 188
IFP session load 343
IMPORT command 190
import file, specifying 57, 118
Indicator Mode
controlling formats 33
controlling output files 31
EXPORT command 155
IMPORT command 192
INDICDATA command 197, 198
limiting display data 100
RECORDMODE command 247
response mode 91
specifying 98, 99
INDICDATA command 197
INDICDATA keyword
EXPORT command 155
IMPORT command 191
input files
controlling 30
specifying 54, 55
interactive mode
defining input and output files 54
LOGON command 206
INTERMEDIATE keyword, SESSION SQLFLAG 282
interrupting BTEQ 81, 130
invoking BTEQ 36

J
JCL, submitting BTEQ batch jobs 52

K
Kanji character set support 351
Korean character set support 351, 354

L
LABEL command 200
LARGEDATAMODE command 201
line numbers in scripts or run files 62
log on requirements 39
logging on 42
logical record length 57
LOGOFF command
described 205
example 42
LOGON command
entering 39
password, security 209
TDPID 207
LOGONPROMPT command 211
LOGONPROMPT keyword, DEFAULTS command 140
LOW keyword, NOTIFY command 216
LRECL
EXPORT command 160
for BTEQ output file 57
for lines printed 342, 343, 344, 347

M
macro
command execution messages 71
creating 66
creating a report 70
creating and executing from a script 72
executing 70
granting EXECUTE privilege 69
including in a script 70
running 71
using 61
with IF...THEN clause 189
MAXERROR command 85, 213
maximum error level 85, 90
maximum number of columns 262
maximum number of rows 262
MEDIUM keyword, NOTIFY command 216
MESSAGEOUT command 214
Micro Teradata Director Program (MTDP) 27
modes of session operation 24
MSG keyword, NOTIFY command 217
multiple BTEQ sessions 45
multiple sessions, unloading data 342

N
network-attached systems
character set support 351
diagram 26
differences from channel-attached systems 349, 350
operating system command 84
NOBOM
   EXPORT command 153
   in MESSAGEOUT command 214
NOEJECT 156
NONE keyword, SESSION SQLFLAG 282
NOSPL keyword, COMPILe command 134
NOTIFY command 216
Notify User Exits 218
NULL command 229
null specification
   changing 96
default 96
number of sessions, setting 45

O
OFF keyword
   ERRORLEVEL command 147
   FORMCHAR command 176
   INDICDATA command 197
   LOGONPROMPT command 201, 211
   NOTIFY command 217
   RECORDMODE command 246
   REPEATSTOP command 257
   RETCANCEl command 260
   OMIT command 232
ON keyword
   ERRORLEVEL command 147
   INDICDATA command 197
   LOGONPROMPT command 201, 211
   RECORDMODE command 246
   REPEATSTOP command 257
   RETCANCEl command 260
OPEN keyword, EXPORT command 156
operating environments 27
operating modes for BTEQ sessions 24
operating system commands 84
OS command 84, 235
output
   controlling 31
default specifications 95
erors 91
   specifying 54
standard errors 59
overview of basic process 26

P
page format
   breaks in reports 104
   folds and breaks 164
   RTITLE and numbers 115
   size in reports 109
PAGEBREAK command 238
PAGELENGTH command 241
parallel BTEQ sessions 45
Parallel Database Extensions (PDE) 27
parcel flavor, IMPORT command 192
parcel length, IMPORT command 192
password
   changing 44
   entering 40
   expired 43
   LOGON command 39
   security 209
PC-DOS
   using output files 116
personnel database, sample 331
printer formats in reports 109
printing output 119
prileges
   client system access 36
   COMPILe command 135
   CREATE MACRO 66
   CREATE PROCEDURE 72, 135
database access, for creating and using macros 66
database for stored procedures 72
DROP 72, 135
EXECUTE 66, 69, 72
for using macros 69
Personnel sample database 334
product version numbers 3, 27

Q
QUEUE keyword, NOTIFY command 217
QUIET command 243
QUIT command 43, 245
quotation marks, general rules for 121

R
Record format, with IMPORT 192, 193
record length 57
Record Mode
   controlling formats 33
   controlling output files 31
   EXPORT command 153, 158, 159
GET command example 160
IMPORT command 192
in export file 118
limiting display data 100
RECORDMODE command 246
response mode 91
specifying 98, 99
RECORDMODE command 246
REMARK command 249
REPEAT command
and = command 253
description 252
examples for using 78
REPEATSTOP command 257
REPORT keyword
EXPORT command 156
IMPORT command 191
with IMPORT 194
REPORTALIGN command 258
reports
blank lines, inserting 302, 304
changing command sequence 115
converting displays to reports 115
dashed lines 319
divert off 116
footers 107
form feed 110
formatting for 101
headers 107
headings 106
horizontal display 102
inserting blank lines 103
macros for creating 70
page breaks 104
page size 109
printer formats 109
printing 117
producing 115
separating columns 105
side-titles 106
splitting lines 105
titles 107, 115
underlining 103
vertical display 101
WITH clause 110
request retry function, disabling 97
request, definition 252
RESET keyword, EXPORT command 156
response mode 91
controlling formats 33
controlling output files 31
EXPORT command 159
INDICDATA command 197
viewing with SHOW-contROLS 159
RETCANCEL command 260
RETLIMIT command 262
RETRY codes, failure 265
RETRY command
description 265
disabling 97
return codes 85
rows
maximum number of 262
RTITLE command 107
description 267
headings, specifying 107
order 115
RUN command 271
run file
displaying results 64
executing 64
revising 65
specifying 59
terminating 64
using 61
S
sample, personnel database 331
script
creating 62
displaying results 63
macros in 70, 72
revising 65
stored procedures in 74, 75
testing and branching 75
using 61
security, passwords 209
SELECT statement
with format phrase 158
with repeats 79
separating columns in reports 105
SEPARATOR command 273
SESSION CHARSET command 276
character set support 353
command line options 37
description 276
SESSION RESPBUFLen command 280
SESSION SQLFLAG command 282
SESSION TRANSACTION command 284
sessions
control commands 28
mode 24, 136
multiple BTEQ 45
overview 23
parallel 45
setting number 45
SESSIONS command 287
SET LARGEDATAMODE command 201
severity levels 85, 90
shell output stream 60
SHOW command 290
SHOW CONTROLS command 282, 284, 289
SHOW ERRORMAP command 295
SHOW VERSIONS command 297
SIDETITLES command 299
side-titles in reports 106
signals, UNIX 52
single-break function 81
single-statement requests 252
SKIP keyword, RUN command 271
SKIPDOUBLE command 302
SKIPLINE command 304
speed, using parallel sessions 45
SPL keyword, COMPILE command 134
SPL option, COMPILE 135
splitting lines in reports 105
SQL requests
continuing vs. terminating 50
determining request type 48
repeating 126
request types 47
statement failure 97
USING modifier with 345
SQLFLAG 282
standard input file 55
starting BTEQ 36
STDERR keyword
channel-attached systems 149
description 149
error output file 92
ERROROUT command 149
overview 59
STDOUT keyword
error output file 92
ERROROUT command 149
network-attached systems 149
NOTIFY command 228
overview 59
UNIX output example 223
stored procedures
command execution 73
compilation errors 85
compiling 91
creating 73
creating and executing 75
errors 91
privileges required 72
replacing 73
using 61
string length, EXPORT command 158
SUN, shell output stream 60
supported releases 3
SUPPRESS command 102, 307
syntax
BTEQ commands 121
variables 123
SYSABEND 53
SYSIN file 39, 53, 54, 55, 61, 62, 63
SYSOUT file 39
SYSPRINT file 56, 64, 65
allocation of 39
defining 54
in batch files 53
limitation for exports 59, 154
macro sample 71
specifying output 56
stored procedure example 74
system access requirements 35
SYSTERM file 53, 60, 92, 149

T
TDP command 310
tdpid
default setting 44
LOGON command 39
TDP command 310
using 207
Teradata Director Program (TDP) or (MTDP) 27
Teradata mode
REPEAT command 255
role in compilation 136
role in compilation errors 136
SESSION TRANSACTION command 284, 285
stored procedure example 75
Teradata server
changing defaults 44
overview 27
Teradata SQL 47
terminating a run file 64
testing and branching commands 75
TIMEMSG command 312
TITLEDASHES command 312, 315
titles
creating with RTITLE 115
specifying for reports 107
titles, changing 236
TRANSACTION command. See SESSION TRANSACTION command
transaction semantics 136
TSO command
description 318
verifying file names 84
underline characters, inserting in reports 103
UNDERLINE command 319
Unicode
    rules 355
    SESSION CHARSET 276, 353
    starting a BTEQ session 36
UNIX
    entering commands 84
    signals 52
    using output files 116
UNKNOWN keyword
    ERRORLEVEL command 147
unlimited return mode 264
userid
    described 39
    entering 40
USING modifier 345
UTF16 355
UTF8 355

VARTEXT keyword, IMPORT command 192
version numbers 3, 27
vertical display of data 101

warnings, stored procedure compilation 91
WIDTH command 322
Windows
    entering commands 85
windows, changing titles of 236
WITH clauses
    identifiers 124
    in reports 110
    SEPARATOR command 274
    SIDETITLES command 299
    TITLEDASHES command 315
WITH keyword, COMPILE command 134
withlist
    defined 124
    SIDETITLES command 299

z/OS
    JCL example 344, 347
    printing output 119
    running BTEQ on 52