



---

# Teradata FastExport

## Reference

Release 13.00.00  
B035-2410-088A  
April 2009

The product or products described in this book are licensed products of Teradata Corporation or its affiliates.

Teradata, BYNET, DBC/1012, DecisionCast, DecisionFlow, DecisionPoint, Eye logo design, InfoWise, Meta Warehouse, MyCommerce, SeeChain, SeeCommerce, SeeRisk, Teradata Decision Experts, Teradata Source Experts, WebAnalyst, and You've Never Seen Your Business Like This Before are trademarks or registered trademarks of Teradata Corporation or its affiliates.

Adaptec and SCSISelect are trademarks or registered trademarks of Adaptec, Inc.

AMD Opteron and Opteron are trademarks of Advanced Micro Devices, Inc.

BakBone and NetVault are trademarks or registered trademarks of BakBone Software, Inc.

EMC, PowerPath, SRDF, and Symmetrix are registered trademarks of EMC Corporation.

GoldenGate is a trademark of GoldenGate Software, Inc.

Hewlett-Packard and HP are registered trademarks of Hewlett-Packard Company.

Intel, Pentium, and XEON are registered trademarks of Intel Corporation.

IBM, CICS, RACF, Tivoli, and z/OS are registered trademarks of International Business Machines Corporation.

Linux is a registered trademark of Linus Torvalds.

LSI and Engenio are registered trademarks of LSI Corporation.

Microsoft, Active Directory, Windows, Windows NT, and Windows Server are registered trademarks of Microsoft Corporation in the United States and other countries.

Novell and SUSE are registered trademarks of Novell, Inc., in the United States and other countries.

QLogic and SANbox are trademarks or registered trademarks of QLogic Corporation.

SAS and SAS/C are trademarks or registered trademarks of SAS Institute Inc.

SPARC is a registered trademark of SPARC International, Inc.

Sun Microsystems, Solaris, Sun, and Sun Java are trademarks or registered trademarks of Sun Microsystems, Inc., in the United States and other countries.

Symantec, NetBackup, and VERITAS are trademarks or registered trademarks of Symantec Corporation or its affiliates in the United States and other countries.

Unicode is a collective membership mark and a service mark of Unicode, Inc.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other product and company names mentioned herein may be the trademarks of their respective owners.

**THE INFORMATION CONTAINED IN THIS DOCUMENT IS PROVIDED ON AN "AS-IS" BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO THE ABOVE EXCLUSION MAY NOT APPLY TO YOU. IN NO EVENT WILL TERADATA CORPORATION BE LIABLE FOR ANY INDIRECT, DIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING LOST PROFITS OR LOST SAVINGS, EVEN IF EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.**

The information contained in this document may contain references or cross-references to features, functions, products, or services that are not announced or available in your country. Such references do not imply that Teradata Corporation intends to announce such features, functions, products, or services in your country. Please consult your local Teradata Corporation representative for those features, functions, products, or services available in your country.

Information contained in this document may contain technical inaccuracies or typographical errors. Information may be changed or updated without notice. Teradata Corporation may also make improvements or changes in the products or services described in this information at any time without notice.

To maintain the quality of our products and services, we would like your comments on the accuracy, clarity, organization, and value of this document. Please e-mail: [teradata-books@lists.teradata.com](mailto:teradata-books@lists.teradata.com)

Any comments or materials (collectively referred to as "Feedback") sent to Teradata Corporation will be deemed non-confidential. Teradata Corporation will have no obligation of any kind with respect to Feedback and will be free to use, reproduce, disclose, exhibit, display, transform, create derivative works of, and distribute the Feedback and derivative works thereof without limitation on a royalty-free basis. Further, Teradata Corporation will be free to use any ideas, concepts, know-how, or techniques contained in such Feedback for any purpose whatsoever, including developing, manufacturing, or marketing products or services incorporating Feedback.

**Copyright © 1998-2009 by Teradata Corporation. All Rights Reserved.**

## Purpose

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

Teradata FastExport is a command-driven utility that uses multiple sessions to quickly transfer large amounts of data from tables and views of the Teradata Database to a client-based application. This book describes the operational features and capabilities of the utility, and includes the syntax for Teradata FastExport commands.

## Audience

This book is intended for use by:

- System and application programmers
- System administrators

## Supported Releases

This book supports the following releases:

- Teradata Database 13.00.00
- Teradata Tools and Utilities 13.00.00
- Teradata FastExport Reference Version 13.00.00

**Note:** See “[Interactive Mode](#)” on page 22 to verify the Teradata FastExport version number.

To locate detailed supported-release information:

- 1 Go to <http://www.info.teradata.com/>.
- 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**.
- 6 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:

- Familiarity with computer technology
- Knowledge of database management systems and utilities that load and retrieve data
- Teradata SQL

## 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.

Date and Release	Description
April 2009 13.00.00	Updated FastExport configuration file information.
August 2008 13.00.00	Changes included for Teradata Tools and Utilities 13.00.00: <ul style="list-style-type: none"><li>• Standalone utilities cannot handle objects that include a semicolon in the object name.</li><li>• Can use “xx1A” characters in Unicode data.</li><li>• Support for Period data type.</li><li>• SET and ACCEPT are valid commands preceding LOGON and LOGTABLE.</li><li>• Updated IBM names to correct references.</li><li>• Removed APPLY-WHERE from IMPORT command syntax diagram.</li><li>• Added UDT/UDM information for FastExport as Appendix D.</li></ul>

## 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, *mmyx* 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 *mmy* of a related publication to the date on the cover of this book. This ensures that the publication selected supports the same release.

Type of Information	Description	Access to Information
Release overview Late information	<p>Use the Release Definition for the following information:</p> <ul style="list-style-type: none"> <li>• Overview of all of the products in the release</li> <li>• Information received too late to be included in the manuals</li> <li>• Operating systems and Teradata Database versions that are certified to work with each product</li> <li>• Version numbers of each product and the documentation for each product</li> <li>• Information about available training and the support center</li> </ul>	<ol style="list-style-type: none"> <li>1 Go to <a href="http://www.info.teradata.com/">http://www.info.teradata.com/</a>.</li> <li>2 Click <b>General Search</b> under <b>Online Publications</b>.</li> <li>3 Type <i>2029</i> in the <b>Publication Product ID</b> box.</li> <li>4 Click <b>Search</b>.</li> <li>5 Select the appropriate Release Definition from the search results.</li> </ol>
Additional product information	<p>Use the Teradata Information Products web site to view or download specific manuals that supply related or additional information to this manual.</p>	<ol style="list-style-type: none"> <li>1 Go to <a href="http://www.info.teradata.com/">http://www.info.teradata.com/</a>.</li> <li>2 Click <b>Data Warehousing</b> under <b>Online Publications, Browse by Category</b>.</li> <li>3 Do one of the following: <ul style="list-style-type: none"> <li>• For a list of Teradata Tools and Utilities documents, click <b>Teradata Tools and Utilities</b>, and then select an item under <b>Releases</b> or <b>Products</b>.</li> <li>• Select a link to any of the data warehousing publications categories listed.</li> </ul> </li> </ol> <p>Specific books related to Teradata FastExport are as follows:</p> <ul style="list-style-type: none"> <li>• <i>Teradata Tools and Utilities Command Summary B035-2401-mmyA</i></li> </ul>
CD-ROM images	<p>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.</p>	<ol style="list-style-type: none"> <li>1 Go to <a href="http://www.info.teradata.com/">http://www.info.teradata.com/</a>.</li> <li>2 Click <b>Data Warehousing</b> under <b>Online Publications, Browse by Category</b>.</li> <li>3 Click <b>CD-ROM List and Images</b>.</li> </ol>
Ordering information for manuals	<p>Use the Teradata Information Products web site to order printed versions of manuals.</p>	<ol style="list-style-type: none"> <li>1 Go to <a href="http://www.info.teradata.com/">http://www.info.teradata.com/</a>.</li> <li>2 Click <b>How to Order</b> under <b>Print &amp; CD Publications</b>.</li> <li>3 Follow the ordering instructions.</li> </ol>

Type of Information	Description	Access to Information
General information about Teradata	<p>The Teradata home page provides links to numerous sources of information about Teradata. Links include:</p> <ul style="list-style-type: none"><li>• Executive reports, case studies of customer experiences with Teradata, and thought leadership</li><li>• Technical information, solutions, and expert advice</li><li>• Press releases, mentions, and media resources</li></ul>	<ol style="list-style-type: none"><li>1 Go to <a href="https://www.teradata.com">Teradata.com</a>.</li><li>2 Select a link.</li></ol>

# Table of Contents

---

<b>Preface</b> .....	3
Purpose .....	3
Audience .....	3
Supported Releases .....	3
Prerequisites .....	4
Changes to This Book.....	4
Additional Information .....	4

---

## **Chapter 1: Overview** .....

FastExport Utility .....	13
Description .....	13
What it Does.....	13
How it Works.....	14
Operating Features and Capabilities .....	14
Operating Modes .....	14
Character Sets.....	14
Task Status Reporting .....	14
FastExport Commands.....	16
FastExport Support Activity Commands.....	16
FastExport Task Activity Commands.....	17
Teradata SQL Statements .....	17
FastExport Example .....	19

---

## **Chapter 2: Using FastExport** .....

Invoking FastExport .....	21
File Requirements .....	21
Interactive Mode .....	22
Batch Mode.....	22
Run-time Parameters.....	23

z/OS Example . . . . .	28
z/VM Example . . . . .	29
UNIX and Windows Examples . . . . .	30
Terminating FastExport . . . . .	31
Normal Termination . . . . .	31
Abort Termination . . . . .	31
Restarting a Paused FastExport Job . . . . .	32
Paused FastExport Jobs . . . . .	32
After a Job Script Error . . . . .	33
After Hardware Failures or Software Error Conditions . . . . .	33
After an AP Reset Condition . . . . .	33
Programming Considerations . . . . .	34
FastExport Configuration File . . . . .	34
Generated MultiLoad Script File . . . . .	36
FastExport Command Conventions . . . . .	38
Variables . . . . .	39
ANSI/SQL DateTime Specifications . . . . .	40
Comments . . . . .	40
Character Set Specification . . . . .	41
Graphic Data Types . . . . .	46
Graphic Constants . . . . .	46
Select Requests . . . . .	46
Restrictions and Limitations . . . . .	47
Termination Control Codes . . . . .	48
UNIX Signals . . . . .	48
Using INMOD, OUTMOD, and Notify Exit Routines . . . . .	49
Overview . . . . .	49
Programming Considerations for Using Routines . . . . .	50
FastExport/INMOD Routine Interface . . . . .	54
FastExport/OUTMOD Routine Interface . . . . .	56
FastExport/Notify Exit Routine Interface . . . . .	57
Writing a FastExport Job Script . . . . .	59
Definition . . . . .	59
Using Checkpoints in a Single Export Job . . . . .	61
<hr/>	
<b>Chapter 3:</b>	
<b>FastExport Commands . . . . .</b>	<b>63</b>
Syntax Notes . . . . .	63
Object Name Restrictions . . . . .	63
Geospatial Data Restrictions . . . . .	63



ACCEPT.....	64
BEGIN EXPORT.....	67
DATEFORM .....	74
DISPLAY .....	75
END EXPORT.....	77
EXPORT .....	78
FIELD.....	87
FILLER.....	94
IF, ELSE, and ENDIF .....	96
IMPORT .....	98
LAYOUT .....	104
LOGDATA.....	107
LOGMECH .....	108
LOGOFF .....	109
LOGON .....	111
LOGTABLE .....	114
ROUTE MESSAGES.....	116
RUN FILE .....	118
SET .....	120
SYSTEM.....	122

---

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

Syntax Diagram Conventions .....	123
Strings .....	125
Multiple Legitimate Phrases .....	127
Sample Syntax Diagram.....	128
Diagram Identifier .....	128

---

## **Appendix B: Invocation Examples .....**

z/VM.....	129
z/OS .....	131
UNIX and Windows .....	135

---

**Appendix C:  
INMOD, OUTMOD and  
Notify Exit Routine Examples**.....139

z/VM.....140  
z/OS .....145  
UNIX .....156  
Windows .....166

---

**Appendix D:  
User-Defined-Types  
and User-Defined-Methods** .....177

User-Defined-Types and User-Defined-Methods .....177  
    User-Defined Types (UDTs).....177  
    User-Defined-Methods (UDMs) .....178  
    Creating UDTs with FastExport.....178  
    Inserting and Retrieving UDTs with Client Products.....178  
    External Types .....178  
    Inserting UDTs with FastExport.....179  
    Retrieving UDTs with FastExport .....179  
    Retrieving UDT Metadata with FastExport.....179

---

**Glossary** .....181

---

**Index**.....207

# List of Tables

Table 1: FastExport Support Activity Commands . . . . .	16
Table 2: FastExport Task Activity Commands . . . . .	17
Table 3: Supported Teradata SQL Statements in FastExport . . . . .	18
Table 4: Data Sets, Files and Devices for Teradata FastExport . . . . .	21
Table 5: Run-time Parameter Descriptions (Channel-Attached Systems) . . . . .	23
Table 6: Run-time Parameter Descriptions (Network-Attached Systems) . . . . .	25
Table 7: System Variables . . . . .	39
Table 8: C Language Comments . . . . .	40
Table 9: Standard Character Sets Supported by FastExport . . . . .	41
Table 10: Site-Defined Character Sets . . . . .	42
Table 11: Methods for Specifying Character Sets . . . . .	44
Table 12: Commands Impacting Multibyte Character Sets . . . . .	45
Table 13: FastExport Programming Restrictions and Limitations . . . . .	47
Table 14: INMOD and OUTMOD Routines . . . . .	49
Table 15: Languages Supported by Platform and Type of User-Developed Routine . . . . .	50
Table 16: Programming Structure for INMOD Routines . . . . .	51
Table 17: Entry Points for INMOD, OUTMOD, and Notify Exit Routines . . . . .	52
Table 18: FastExport-to-INMOD Status Codes . . . . .	54
Table 19: INMOD-to-FastExport Interface Status Codes . . . . .	55
Table 20: FastExport-to-OUTMOD Interface Entry Codes . . . . .	56
Table 21: Events Passed to the Notify Exit Routine . . . . .	58
Table 22: Commands for Establishing FastExport Support Environment . . . . .	60
Table 23: Commands for Specifying the FastExport Task . . . . .	60
Table 24: ACCEPT Command Usage Notes . . . . .	65
Table 25: Events That Create Notifications . . . . .	71
Table 26: BEGIN EXPORT Usage Notes . . . . .	72
Table 27: DATEFORM Command Usage Notes . . . . .	74
Table 28: DISPLAY Command Usage Notes . . . . .	75
Table 29: END EXPORT Command Usage Notes . . . . .	77
Table 30: EXPORT Command Usage Notes . . . . .	81
Table 31: Record Length and Block Size Specifications (Channel-Attached Client Systems) . . . . .	83

Table 32: Data Type Descriptions (Channel-Attached Client Systems) . . . . .	84
Table 33: FIELD Command Usage Notes . . . . .	88
Table 34: ANSI/SQL DateTime Specifications . . . . .	89
Table 35: FILLER Command Usage Notes . . . . .	95
Table 36: IF, ELSE and END IF Command Usage Notes . . . . .	96
Table 37: IMPORT Command Usage Notes . . . . .	102
Table 38: LAYOUT Command Usage Notes . . . . .	105
Table 39: LOGOFF Command Usage Notes . . . . .	109
Table 40: LOGON Command Usage Notes . . . . .	112
Table 41: LOGTABLE Command Usage Notes . . . . .	114
Table 42: ROUTE MESSAGES Command Usage Notes . . . . .	117
Table 43: RUN FILE Command Usage Notes . . . . .	119
Table 44: SET Command Usage Notes . . . . .	120

---

This chapter provides an introductory overview of the Teradata FastExport utility. Topics include:

- [FastExport Utility](#)
- [Operating Features and Capabilities](#)
- [FastExport Commands](#)
- [FastExport Example](#)

## FastExport Utility

This section provides a general description of FastExport, what it does, and how it works.

### Description

FastExport is a command-driven utility that uses multiple sessions to quickly transfer large amounts of data from tables and views of the Teradata Database to a client-based application.

Data can be exported from any table or view where the SELECT access privilege has been granted. The destination for the exported data can be:

- A file on a channel-attached or network-attached client system
- An Output Modification (OUTMOD) routine written to select, validate, and preprocess the exported data

**Note:** Full tape support is not available for any function in FastExport for network-attached client systems. To export data to a tape, write a custom access module that interfaces with the tape device. For information about how to write a custom access module, see the *Teradata Tools and Utilities Access Module Programmer Guide*.

### What it Does

When FastExport is invoked, the utility executes the FastExport commands and Teradata SQL statements in the FastExport job script. These direct FastExport to:

- 1 Log on to the Teradata Database for a specified number of sessions, using username, password, and tdpid/acctid information
- 2 Retrieve the specified data from the Teradata Database, in accordance with format and selection specifications

- 3 Export the data to the specified file or OUTMOD routine on a client system
- 4 Log off the Teradata Database

## How it Works

FastExport processes a series of FastExport commands and Teradata SQL statements entered, usually as a batch mode job script.

The FastExport commands provide the session control and data handling specifications for the data transfer operations. The Teradata SQL statements perform the actual data export functions on the Teradata Database tables and views.

# Operating Features and Capabilities

This section describes the key operational capabilities for running the FastExport utility. For specific information on supported operating systems, refer to *Teradata Tools and Utilities ###,###,## Supported and Certified Versions*, B035-3119-*mmm*K. This spreadsheet shows version numbers and platform information for all Teradata Tools and Utilities products and is available at <http://www.info.teradata.com/>

## Operating Modes

FastExport runs in the following operating modes:

- Interactive
- Batch

## Character Sets

FastExport supports the character sets shipped with Teradata; this includes Latin, Japanese, Chinese, and Korean character sets, along with ASCII, EBCDIC, UTF-8, and UTF-16. For additional information about character-set support and definition, see “[Character Set Specification](#)” on page 41.

## Task Status Reporting

FastExport utility has three ways to provide information about the status of jobs that are still in progress and those that have just completed:

- Logon/connect messages
- Operational status messages
- Logoff/disconnect messages

Additionally, the Query Session utility (QrySessn) provides real-time, phase-oriented progress reports at selected intervals during the FastExport job.

The FastExport utility writes messages to either:

- The customary output destination for a client system (SYSPRINT/stdout or the redirected stdout)  
or
- An alternate destination specified in a ROUTE MESSAGES command

The utility also writes operational status information in the restart log table so it can be restored after a system restart operation.

The following sections describe reporting methods available for monitoring FastExport.

### **Logon/Connect Messages**

In addition to input command directives (except for the logon password), FastExport lists the options specifications for each task:

- SESSIONS *limit*
- TENACITY *hours*
- SLEEP *minutes*

### **Operational Status Messages**

During the progress of a FastExport job, the utility displays a message each time a SELECT statement:

- Executes
- Completes

The completion message also indicates the total number of data blocks generated by the statement.

Also, at five-minute intervals, FastExport reports:

- The total number of records that have been exported to the output file
- The number of blocks processed for each executing minute, displayed as both a running average and a five-minute average

Turn off status messages by setting:

- The run-time parameter `-s` to OFF
- The FastExport CONFIG FILE entry set to Status=OFF

For a detailed description, see [“Run-time Parameters” on page 23](#) and [“FastExport Configuration File” on page 34](#).

### **Logoff/Disconnect Messages**

Issued in response to the LOGOFF command, the FastExport logoff/disconnect message lists:

- The time that the LOGOFF command was executed
- Whether the disconnect operation was successful
- Whether the restart log table was dropped or kept, depending on the success or failure of the job
- The total processor time used

- The job start/end time and date
- The highest return code encountered by the job

### Query Session Utility

QrySessn is a separate utility that monitors the progress of a FastExport job on the Teradata Database. QrySessn reports status information for each phase of the FastExport job.

Execute the Query Session utility from either:

- A system console using the Database Window (DBW) interface on a Teradata Database for UNIX system, or
- Teradata Manager

For complete information about using the Query Session utility, see the QrySessn chapter in the *Utilities* reference documentation.

## FastExport Commands

FastExport provides commands for support and task activities and supports a subset of Teradata SQL statements.

### FastExport Support Activity Commands

Support commands establish the FastExport sessions with the Teradata Database and define the operational *support environment* for the FastExport utility. Established support environment options remain in effect until another support command changes them. Support commands do not specify a FastExport task.

[Table 1](#) lists the FastExport commands that perform support activities.

Table 1: FastExport Support Activity Commands

FastExport Command	Function
<a href="#">ACCEPT</a>	Allows the value of one or more utility variables to be accepted from either a file or an environment variable
<a href="#">DATEFORM</a>	Specifies the form of the DATE data type specifications for the FastExport job
<a href="#">DISPLAY</a>	Writes messages to the specified destination
<a href="#">ELSE</a> (see <a href="#">IF</a> , <a href="#">ELSE</a> , and <a href="#">ENDIF</a> )	Introduces commands and statements that execute when a preceding IF condition is false
<a href="#">ENDIF</a> (see <a href="#">IF</a> , <a href="#">ELSE</a> , and <a href="#">ENDIF</a> )	Delimits the group of FastExport commands that were subject to previous IF or ELSE conditions
<a href="#">IF</a> (see <a href="#">IF</a> , <a href="#">ELSE</a> , and <a href="#">ENDIF</a> )	Introduces a conditional expression whose value initiates execution of subsequent commands
<a href="#">LOGOFF</a>	Disconnects all active sessions and terminates FastExport



Table 1: FastExport Support Activity Commands (continued)

FastExport Command	Function
LOGON	Specifies the LOGON string to be used in connecting all sessions established by FastExport
LOGTABLE	Specifies a restart log table for the FastExport checkpoint information
ROUTE MESSAGES	Identifies an alternate destination of FastExport output messages
RUN FILE	Invokes the specified external file as the current source of utility commands and Teradata SQL statements
SET	Assigns a data type and a value to a utility variable
SYSTEM	Suspends operation of FastExport and executes any valid local operating system command

## FastExport Task Activity Commands

Task commands specify the actual processing that takes place for each FastExport task.

Table 2 lists the FastExport commands that perform task activities.

Table 2: FastExport Task Activity Commands

FastExport Command	Function
BEGIN EXPORT	Signifies the beginning of an export task and sets the specifications for the task sessions with the Teradata Database
END EXPORT	Signifies the end of an export task and initiates processing by the Teradata Database
EXPORT	Provides: <ul style="list-style-type: none"> <li>The client system destination and file format specifications for the export data retrieved from the Teradata Database</li> <li>A generated MultiLoad script file which can later reload the export data back into the Teradata Database</li> </ul>
FIELD	Specifies a field of the input record that provides data values for the constraint parameters of the SELECT statement
FILLER	Specifies a field of the input record that will not be sent to the Teradata Database as part of the input record that provides data values for the constraint parameters of the SELECT statement
IMPORT	Defines the client file that provides the USING data values for the SELECT statement
LAYOUT	Specifies, in conjunction with an immediately following sequence of FIELD and FILLER commands, the layout of the file that provides data values for the USING modifier of the SELECT statement

## Teradata SQL Statements

Teradata SQL statements define and manipulate the data stored in the Teradata Database.

FastExport supports a subset of Teradata SQL statements. As a result, other utilities do not have to be invoked to perform routine database maintenance functions before executing FastExport utility tasks. For example, the supported Teradata SQL statements can be used to:

- Create or modify the table to export from
- Establish a database as an explicit table-name qualifier
- Add checkpoint specifications to a journal table

**Note:** The following restrictions apply to Teradata SQL statements in FastExport job scripts:

- FastExport supports *only* the Teradata SQL statements listed in [Table 3](#)
- Except for the SELECT statement, the supported Teradata SQL statements *must* appear either before or after an export task specification—they cannot appear between the BEGIN EXPORT and the END EXPORT commands of an export task
- The SELECT statement is supported *only* within an export task specification—it must appear after the BEGIN EXPORT command and before the END EXPORT command of an export task

Teradata SQL statements supported by Teradata FastExport are listed in [Table 3](#). To use other Teradata SQL statements, exit FastExport and enter them from another application, such as Basic Teradata Query (BTEQ).

Table 3: Supported Teradata SQL Statements in FastExport

Teradata SQL Statement	Function
ALTER TABLE	Changes the column configuration or options of an existing table
CHECKPOINT	Adds a checkpoint entry to a journal table
COLLECT STATISTICS	Collects statistical data for one or more columns of a table
COMMENT	Stores or retrieves comment string associated with a database object
CREATE DATABASE CREATE MACRO CREATE TABLE CREATE VIEW	Creates a new database, macro, table, or view
DATABASE	Specifies a new default database for the current session
DELETE	Removes rows from a table
DELETE DATABASE	Removes all tables, views, and macros from a database
DROP DATABASE	Drops the definition for an empty database from the Data Dictionary
DROP TABLE	Removes a table from the Teradata Database
GIVE	Transfers ownership of a database to another user
GRANT	Grants access privileges to a database object
INSERT	Inserts new rows to a table
MODIFY DATABASE	Changes the options of an existing database

Table 3: Supported Teradata SQL Statements in FastExport (continued)

Teradata SQL Statement	Function
RENAME	Changes the name of an existing table, view, or macro
REPLACE MACRO REPLACE VIEW	Redefines an existing macro or view
REVOKE	Rescinds access privileges to a database object
SET QUERY_BAND	Allows a set of name-value pairs to be defined by the user and/or middle tier application so they can be customized to each application's unique needs
SET SESSION COLLATION	Overrides the collation specification for the current session
UPDATE	Changes the column values of an existing row in a table

For syntax and a complete description of each Teradata SQL statement, see *SQL Data Definition Language* and *SQL Data Manipulation Language*.

## FastExport Example

The following FastExport job script example executes a single SELECT statement and returns the results to a data set on the client system:

```
.LOGTABLE utillog ;                /* define restart log          */
.
.LOGON tdpz/user,pswd ;            /* DBC logon string           */
.
.BEGIN EXPORT                      /* specify export function     */
  SESSIONS 20;                    /* number of sessions to be used */
.
.LAYOUT UsingData ;                /* define the input data       */
  .FIELD ProjId * Char(8) ;        /* values for the SELECT       */
  .FIELD WkEnd * Date ;            /* constraint clause.         */
.
.IMPORT INFILE ddname1             /* identify the file that      */
  LAYOUT UsingData ;              /* contains the input data     */
.
.EXPORT OUTFILE ddname2 ;          /* identify the destination    */
  /* file for exported data    */
.
SELECT EmpNo, Hours FROM CHARGES /* provide the SQL SELECT     */
  WHERE WkEnd = :WkEnd           /* statement with values      */
  AND Proj_ID = :ProjID         /* provided by the IMPORT     */
  ORDER BY EmpNo ;              /* command                    */
.
.END EXPORT ;                      /* terminate the export       */
  /* operation                  */
.
.LOGOFF ;                          /* disconnect from the DBS    */
```



---

This chapter provides detailed information about using the FastExport utility. Topics include:

- [Invoking FastExport](#)
- [Terminating FastExport](#)
- [Restarting a Paused FastExport Job](#)
- [Programming Considerations](#)
- [Using INMOD, OUTMOD, and Notify Exit Routines](#)
- [Writing a FastExport Job Script](#)
- [Using Checkpoints in a Single Export Job](#)

## Invoking FastExport

This section describes file requirements, syntax, and run-time parameters for invoking FastExport.

### File Requirements

In addition to the output data destination, FastExport accesses four different data sets/files or input/output devices. [Table 4](#) lists Data Sets, Files and Devices for Teradata FastExport.

Table 4: Data Sets, Files and Devices for Teradata FastExport

Data Set/File or Device	Provides
Standard input	FastExport commands and Teradata SQL statements that make up a FastExport job
Standard output	Destination for FastExport output responses and messages
Standard error	Destination for FastExport errors
Configuration	Optional specification of FastExport utility default values

When running FastExport in interactive mode, the terminal keyboard functions as the standard input device and the display screen is the standard output/error device. When running FastExport in batch mode, a data set or file name must be specified for each of these functions. The method of doing this varies, depending on the configuration of the client system:

- On network-attached client systems, use the standard redirection mechanism (< *infile* and > *outfile*) to specify the FastExport files when invoking the utility.
- On channel-attached client systems, use standard z/VM EXEC or z/OS JCL control statements (FILEDEF and DD) to allocate and create the FastExport data sets or files before invoking the utility.

**Note:** On z/OS client systems, the export file should not be a member of a partitioned data set (PDS). If it is, and a Teradata Database or client system failure interrupts the FastExport job, the restart operation will fail.

## Interactive Mode

To invoke FastExport in interactive mode, enter **fexp** at the system command prompt.

fexp

FastExport displays the following message to begin an interactive session:

```
=====
=          FastExport Utility      Release FEXP.mm.mm.mm.mmm      =
=          Platform xxxx          =
=====
=
=      Copyright 1990-2008, Teradata Corporation. ALL RIGHTS RESERVED.=
=
=====
```

where *mm.mm.mm.mmm* is the release level of the FastExport utility software.

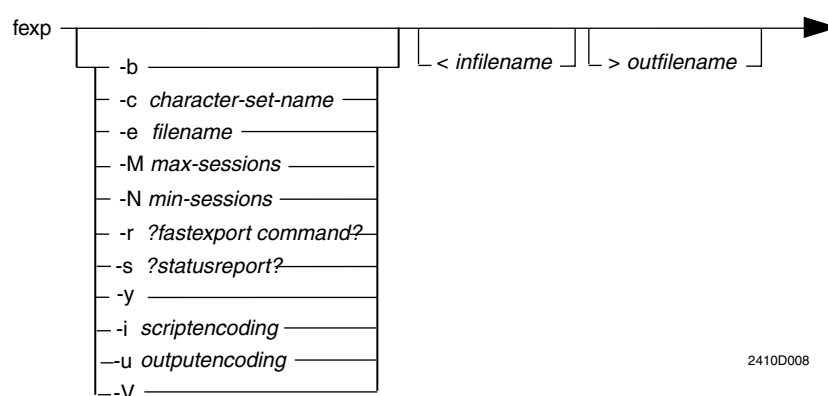
## Batch Mode

This section covers invoking FastExport in batch mode on network-attached and channel-attached client systems.

For a description of how to read syntax diagrams used in this book, see [Appendix A: “How to Read Syntax Diagrams.”](#)

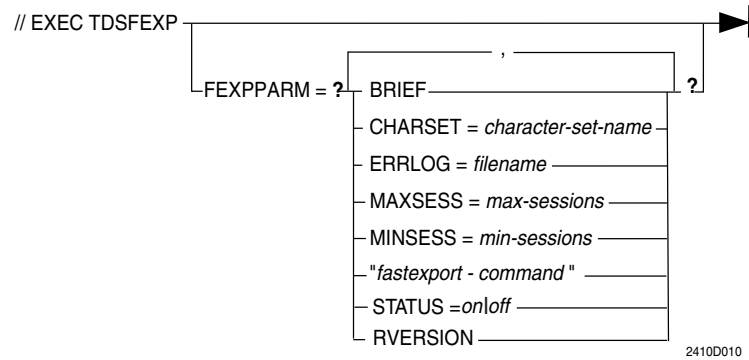
### Batch Mode on Network-Attached Systems

To invoke FastExport in batch mode on network-attached client systems, see the run-time parameter descriptions in [Table 6 on page 25](#) and use the following syntax:



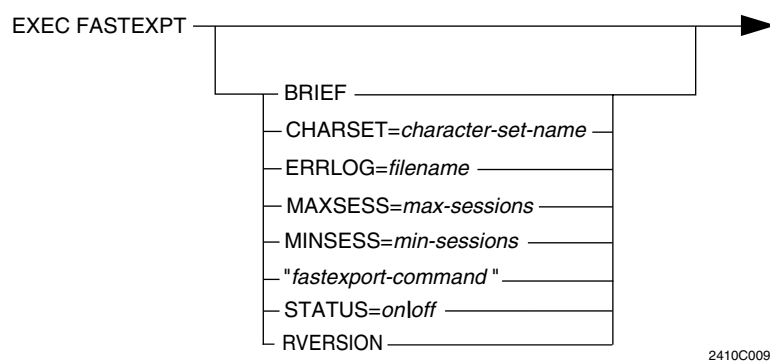
## Batch Mode on Channel-Attached z/OS Systems

To invoke FastExport in batch mode on channel-attached z/OS client systems, see the run-time parameter descriptions in [Table 5 on page 23](#) and use the following syntax.



## Batch Mode on Channel-Attached z/VM Systems

To invoke FastExport in batch mode on channel-attached z/VM client systems, see the run-time parameter descriptions in [Table 5 on page 23](#) and use the following syntax.



**Note:** On z/VM, use the following statement before the EXEC FASTEXPT statement:

```
"GLOBAL LOADLIB DYNAMIC"
```

## Run-time Parameters

[Table 5](#) lists FastExport run-time parameters for channel-attached configurations.

Table 5: Run-time Parameter Descriptions (Channel-Attached Systems)

Parameter	Description
BRIEF	Reduced print output run-time parameter that limits FastExport printout to the least information required to determine the success or failure of the job

Table 5: Run-time Parameter Descriptions (Channel-Attached Systems) (continued)

Parameter	Description
CHARSET= <i>character-set-name</i>	<p>Character set specification for the FastExport job</p> <p>A character set name can be specified. The character set specification remains in effect for the entire FastExport job, even if the Teradata Database resets, causing the FastExport job to be restarted.</p> <p><b>Caution:</b> The character set specification does not remain in effect if the client system fails, or if the FastExport job is cancelled. In these cases, when resubmitting the job, use the same character set specification which was used on the initial job. If a different character set specification is used when a job is resubmitted, the data loaded by the restarted job does not appear the same as the data loaded by the initial job.</p> <p>If a character set specification is not entered, then FastExport uses the specifications in the CLIV2 default files: HSHSPB for channel-attached client systems.</p> <p>Otherwise, the default is whatever character set is specified for the Teradata Database whenever FastExport is invoked.</p> <p><b>Note:</b> When using UTF8 client character set on the mainframe, the client character set name needs to be specified by the runtime parameter (that is, CHARSET=UTF8).</p> <p><b>Note:</b> For other ways to specify the character set, see <a href="#">“Character Set Specification” on page 41</a>.</p>
ERRLOG= <i>filename</i>	<p>Alternate file specification for FastExport error messages</p> <p>On channel-attached client systems, the alternate file specification is limited to eight characters and:</p> <ul style="list-style-type: none"> <li>• On z/OS, it must be a DDNAME defined in the JCL</li> <li>• On z/VM, it must be an existing file definition (FILEDEF)</li> </ul> <p>There is no default error log <i>filename</i> specification.</p>
<i>“fastexport command”</i>	<p>Invocation option that can signify the start of a FastExport job</p> <p>It is usually a RUN FILE command that specifies the file containing the FastExport job script since only one command can be specified.</p> <p>Use the FILEDEF or DD control statements to specify the input and output files before invoking the utility.</p>
MAXSESS = <i>max-sessions</i>	<p>Maximum number of FastExport sessions logged on to the Teradata Database</p> <p>Maximum specification must be greater than zero and no more than the total number of AMPs on the system.</p> <p>Default is one session for each AMP.</p>
MINSESS = <i>min-sessions</i>	<p>Minimum number of FastExport sessions required to run the job</p> <p>Minimum specification must be greater than zero.</p> <p>Default is 1.</p>



Table 5: Run-time Parameter Descriptions (Channel-Attached Systems) (continued)

Parameter	Description
RVERSION	Display version number and stop
STATUS ON OFF	Five-minute status reporting Use <i>ON</i> if enabled (the default) or <i>OFF</i> if disabled.

Table 6 lists FastExport run-time parameters for network-attached configurations.

Table 6: Run-time Parameter Descriptions (Network-Attached Systems)

Parameter	Description
< <i>infile</i>	Name of the standard input file that contains the FastExport commands and Teradata SQL statements on network-attached client systems  The <i>infile</i> specification redirects the standard input ( <i>stdin</i> ). If an <i>infile</i> specification is not entered, the default is <i>stdin</i> .
> <i>outfile</i>	Name of the standard output file for FastExport messages on network-attached systems  The <i>outfile</i> specification redirects the standard output ( <i>stdout</i> ). If an <i>outfile</i> specification is not entered, the default is <i>stdout</i> .  <b>Caution:</b> If an <i>outfile</i> specification is used to redirect <i>stdout</i> , do not use the same <i>outfile</i> as an output or echo destination in the ROUTE MESSAGES command. Doing so produces incomplete results because of the conflicting write operations to the same file.
-b	Reduced print output run-time parameter that limits the FastExport printout to the least information required to determine the success or failure of the job

Table 6: Run-time Parameter Descriptions (Network-Attached Systems) (continued)

Parameter	Description
<p><i>-c character-set-name</i></p>	<p>Character set specification for the FastExport job</p> <p>A character set name can be specified. A character set specification remains in effect for the entire FastExport job, even if the Teradata Database resets, causing the FastExport job to be restarted.</p> <p><b>Caution:</b> The character set specification does not remain in effect if the client system fails, or if the FastExport job is cancelled. In these cases, when resubmitting the job, the same character set specification must be used that was used on the initial job. If a different character set specification is used when a job is resubmitted, the data loaded by the restarted job does not appear the same as the data loaded by the initial job.</p> <p>If a character set specification is not entered, then FastExport uses the specifications in the CLIV2 default files: <i>clisp.dat</i> for network-attached client systems.</p> <p>Otherwise, the default is whatever character set is specified for the Teradata Database whenever you invoke FastExport.</p> <p><b>Note:</b> When using UTF16 client character set on the network, the client character set name needs to be specified by the run-time parameter (that is, <i>-c UTF16</i>).</p> <p><b>Note:</b> For other ways to specify the character set, see <a href="#">“Character Set Specification” on page 41</a>.</p>
<p><i>-e filename</i></p>	<p>Alternate file specification for FastExport error messages</p> <p>Specifying an alternate file name produces a duplicate record of all FastExport error messages; the entire output stream does not have to be viewed to determine why a job failed.</p> <p>There is no default error log <i>filename</i> specification.</p>

Table 6: Run-time Parameter Descriptions (Network-Attached Systems) (continued)

Parameter	Description
<i>-i scriptencoding</i>	<p>Encoding form of the job script</p> <p>The parameter is introduced for use with the UTF16 client character set, so it is only valid when UTF16 client character set is used. If the client character set being used is not UTF16 and the parameter is specified, FastExport reports an error and terminates.</p> <p>The valid values are UTF8, UTF16, UTF16-BE, and UTF16-LE.</p> <ul style="list-style-type: none"> <li>• UTF8 indicates the job script is in UTF8 character set</li> <li>• UTF16 indicates the job script is in UTF16 character set without specifying the endianness</li> <li>• UTF16-BE indicates the job script is in big endian UTF16 character set</li> <li>• UTF16-LE indicates the job script is in little endian UTF16 character set. Or, if UTF16-LE is specified but the UTF-16 Byte Order Mark (BOM) in the script file indicates the script is in big endian, FastExport reports an error and terminates.</li> </ul> <p>The UTF16 or UTF8 BOM can be present or absent in the script file. Specify the input script format with <i>-i</i> runtime parameter (mandatory) and specify session character set with <i>-c</i> runtime parameter (mandatory) to ensure that the BOM in the script file is processed correctly.</p> <p>When UTF16 is specified and the UTF-16 BOM is present in the script file, FastExport will interpret the script according to the endianness indicated by the UTF-16 BOM. When the UTF16 BOM is not present, FastExport will interpret the script according to the endianness indicated by the option value. If the endianness is not indicated by the option value (that is, UTF16 is specified instead of UTF16-BE or UTF16-LE), FastExport will interpret the job script in UTF16 according to the endianness of the client system where the FastExport job invoked. The specified encoding character set applies to all script files included by the .RUN FILE commands.</p> <p>When this runtime parameter is not specified and UTF16 client character is used, FastExport will interpret the job script in UTF16. When UTF8 is specified, FastExport will interpret the job script in UTF8 and will convert SQL and DML statements in the script from UTF8 to UTF16 before sending the SQL and DML statements to Teradata Database.</p>
<i>-M max-sessions</i>	<p>Maximum number of FastExport sessions logged on to the Teradata Database</p> <p>Maximum specification must be greater than zero and no more than the total number of AMPs on the system.</p> <p>Default is one session for each AMP.</p>
<i>-N min-sessions</i>	<p>Minimum number of FastExport sessions required to run the job.</p> <p>Minimum specification must be greater than zero.</p> <p>Default is 1.</p>

Table 6: Run-time Parameter Descriptions (Network-Attached Systems) (continued)

Parameter	Description
<i>-r 'fastexport command'</i>	Invocation option that can signify the start of a FastExport job It is usually a RUN FILE command that specifies the file containing the FastExport job script since only one command can be specified.
<i>-s value</i>	Five-minute status reporting For <i>value</i> , use <i>ON</i> if reporting is enabled, or <i>OFF</i> if disabled.
<i>-u outputencoding</i>	Specifies the encoding form of the job output The parameter is introduced for being used for UTF16 client character set so it is only valid when UTF16 client character set is used. If the client character set being used is not UTF16 and the parameter is specified, FastExport reports an error and terminates. The valid values are UTF16-BE, UTF16-LE, and UTF16: <ul style="list-style-type: none"> <li>• UTF16-BE instructs FastExport to print the job output in the big endian UTF16 character set</li> <li>• UTF16-LE instructs FastExport to print the job output in the little endian UTF16 character set</li> <li>• On big endian client systems, UTF16 instructs FastExport to print the job output in big endian UTF16 character set</li> <li>• On the little endian client systems, UTF16 instructs FastExport to print the job out in little endian UTF16 character set</li> </ul> When the parameter is being used, it should be placed in front of the other runtime parameters to ensure the whole job output will be printed in the desired encoding form. If not placed ahead of the other runtime parameters when invoking the job, a warning message will be printed. When the parameter is not specified and the client character set being used is UTF16, the job output will be printed as UTF16.
<i>-y</i>	Specification for the data encryption option When specified at run time, all sessions will be encrypted.
<i>-V</i>	Display version number and stop

**Note:** For sample JCL and commands that invoke FastExport on the different operating system platforms, see [Appendix B: “Invocation Examples.”](#)

## z/OS Example

The following procedure invokes FastExport on a channel-attached z/OS client system:

```
//FEXPRUN JOB 1, 'FASTEXPORT',
//          MSGCLASS=X,
//          NOTIFY=FEXP,
//*        TYPRUN=SCAN,
//          CLASS=A
/*ROUTE PRINT TSO
```

```

//*****
//*
//*          FASTEXPORT PROC
//*
//*****
//*
//FEXP      PROC PRM=, INPUT=
//FEXPSTP   EXEC PGM=XPORT, PARM='&PRM'
//STEPLIB  DD  DISP=SHR, DSN=TERADATA.APPLoad
//          DD  DISP=SHR, DSN=TERADATA.TRLOAD
//SYSPRINT DD  SYSOUT=*
//OUTFILE  DD  DCB=(NCP=20, RECFM=VB, LRECL=1024, BLKSIZE=32009),
//SYSIN    DD  DISP=SHR, DSN=&INPUT
//          PEND
//*****
//*
//*          RUN FASTEXPORT
//*
//*****
//FEXPSTP   EXEC PROC=FEXP
//FEXPSTP.SYSIN DD  DATA, DLM=###
.
.
.
FastExport commands
.
.
.
//OUTPUT DD SYSOUT=*

```

**Note:** FastExport does not automatically block output records. If the output need to be blocked:

- Specify blocked data (for example, FB) in the record format (RECFM) parameter of the data control block (DCB)
- Specify the block size in the BLOCKSIZE parameter of the DCB

To enhance I/O performance (at the cost of increased storage), increase the value of the network control program (NCP) parameter on an output DCB in JCL. The maximum value for NCP is 99. The additional storage required is the NCP value multiplied by the block size of the output device.

## z/VM Example

The following EXEC procedure invokes FastExport on a channel-attached z/VM client system:

```

/* Run the FastExport program on z/VM.    */
/* Accepts input from terminal (sysin) */
/* Sends output to terminal (sysprint) */
/*                                     */
/* Before running this EXEC, the user   */
/* must issue the z/VM LINK and ACCESS  */
/* commands to link to the CLI         */
/* minidisk and to link to the Sas C   */
/* runtime minidisk.                   */
/*                                     */
"GLOBAL LOADLIB DYNAMC"

```

```
"GLOBAL TXTLIB CLI"  
"GLOBAL LOADLIB LSCTRL"  
"CP SET TIMER REAL"  
"FILEDEF SYSPRINT TERMINAL (LRECL 84 RECFM V"  
"FILEDEF SYSIN     TERMINAL (LRECL 84 RECFM V"  
"XPORT"  
exit rc
```

## UNIX and Windows Examples

The following are examples of four ways to invoke FastExport on network-attached UNIX and Windows client systems:

- `fexp < /home/fexpuser/tests/test1  
> /home/fexpuser/tests/out1`

This command specifies both an input file and an output file:

- `/home/fexpuser/tests/test1` is the input file that provides the FastExport job script
- `/home/fexpuser/tests/out1` is the destination file for output data
- `fexp < /home/fexpuser/tests/test1`

This command specifies only an input file. In this case, the output is written to the standard output device, which is usually a terminal.

- `fexp`

This command specifies neither an input nor an output file. In this case, the terminal provides both the command input and output data destination.

- `fexp -r '.RUN FILE exp.startup;'`

**Note:** Single quotes surrounding the above command line are only valid in a UNIX environment. Double quotes surrounding the above command line are valid in both the UNIX and Microsoft Windows environments.

This command uses the `-r` invocation option to specify the FastExport RUN FILE command. In this case, the FastExport job script is in the `exp.startup` file.

# Terminating FastExport

There are two ways to terminate FastExport:

- Normal Termination
- Abort Termination

Either way ends the FastExport sessions and logs off the Teradata Database. A normal termination, however, does so in an orderly, controlled fashion, and returns messages indicating the status of the FastExport job. An abort termination does not.

## Normal Termination

Use the FastExport LOGOFF command in a FastExport job script to terminate the utility normally on both network-attached and channel-attached client systems:

```
.LOGOFF _____ ; ─▶
                └── retcode ─┘
```

2409A033

FastExport logs off all sessions with the Teradata Database and returns a status message indicating:

- The total processor time that was used
- The job start and stop date/time
- The highest return code that was encountered:
  - 0 if the job completed normally
  - 4 if a warning condition occurred
  - 8 if a user error occurred
  - 12 if a fatal error occurred
  - 16 if no message destination is available

FastExport also:

- Either maintains or drops the restart log table, depending on the success or failure of the job
- If specified, returns the optional *retcode* value to the client operating system

For more information about return codes and the conditions that maintain or drop the restart log table, see [“LOGOFF” on page 109](#).

## Abort Termination

The procedure for aborting a FastExport job depends on whether the utility is running on a network-attached or a channel-attached client system.

---

**To abort a FastExport job running on a network-attached client system:**

- ✓ Press the `Control + C` key combination three times on the workstation keyboard.

---

**To abort a FastExport job running on a channel-attached client system:**

- ✓ Cancel the job from the client system console.

Whenever a FastExport job terminates abnormally:

- The processing of any associated `SELECT` statement is also terminated, and its database access locks are released
- If the Teradata Database has assembled export data in response to a `SELECT` statement, the spool table containing the export data is deleted
- The restart log table is not dropped from the Teradata Database

After aborting a FastExport job, either:

- Restart the job and allow it to run to completion. This, in most cases, is the preferred alternative.  
or
- Drop the restart log table from the Teradata Database. This alternative requires:
  - Restarting the entire job, from the beginning, as a complete new job  
or
  - Abandoning the job, completely

In the case of a FastExport job with only one select request, the results of the two alternatives are essentially the same, because aborting the single select request effectively aborts the entire job. The significant difference occurs when the FastExport job has multiple export tasks, or if it uses an input file that generates multiple select requests.

## Restarting a Paused FastExport Job

This section describes restarting FastExport jobs that have been paused or interrupted.

### Paused FastExport Jobs

A paused FastExport job is one that terminates abnormally, without dropping the restart log table from the Teradata Database. The paused condition can be intentional, or the result of a system failure or error condition.

A FastExport job can be paused intentionally by using the “[Abort Termination](#)” procedure described earlier in this chapter.

Unintentional conditions that can pause a FastExport job include:

- A FastExport job script error



- A hardware failure or software error condition
- An application processor (AP) reset condition

FastExport automatically restarts some paused jobs. Others must be manually restarted. The following subsections describe the manual restart procedure and the factors that affect FastExport restart operations under the different pause conditions.

To manually restart a paused FastExport job, resubmit the entire FastExport job script, using the same restart log table specification. The FastExport utility:

- 1 Reestablishes sessions with the Teradata Database
- 2 Reads the restart log table to determine the restart point
- 3 Resumes processing the FastExport job script

### After a Job Script Error

When FastExport encounters an error in a job script, it generates a diagnostic error message and stops with a nonzero return code. At this point, the script can be modified to correct the error and resubmit the FastExport job. The utility resumes processing at the statement following the last one that completed successfully.

**Note:** When correcting script errors, make changes at or after the indicated error. FastExport does not repeat the commands that executed successfully, but the job will fail, with additional error messages, if the utility detects changes before the indicated error.

### After Hardware Failures or Software Error Conditions

FastExport restarts automatically after system recovery from the following types of hardware failures and software error conditions:

- Down AMP
- CLIV2 error on a client system
- Network failure
- Nonrecoverable I/O error
- Teradata Database restart

If the failure occurred while FastExport was processing a job script with a *single* select request, then the utility resumes processing after system recovery by resubmitting the one select request.

If the failure occurred while FastExport was processing a job script with *multiple* select requests, then the utility resumes processing after system recovery by resubmitting the last select request that was submitted before the failure occurred.

### After an AP Reset Condition

When a FastExport job is interrupted by a resetting AP on the Teradata Database, the restart response depends on the environment in which the utility is running:

- On the resetting AP

- On a nonresetting AP
- On a channel-attached client system

If an AP reset condition occurs and FastExport is running on a *resetting* AP, then the FastExport job is halted and must be manually restarted.

If an AP reset condition occurs and FastExport is running on a *nonresetting* AP, then the FastExport job may or may not be halted, depending on whether it has sessions connected through the resetting AP:

- If the FastExport job has sessions connected through the resetting AP, the utility automatically:
  - Logs off all sessions
  - Logs them back on
  - Rolls back to the most recent checkpoint
  - Resumes processing
- If the FastExport job *does not* have sessions connected through the resetting AP, the utility is not affected by the AP reset condition.

The increased session loading caused by the reconnection of other sessions through the resetting AP may degrade the system response time.

If an AP reset condition occurs and FastExport is running on a channel-attached client system with AP reset containment enabled, then the FastExport job is halted, but does not need to be manually restarted. FastExport automatically:

- Logs off all sessions
- Logs them back on
- Rolls back to the most recent checkpoint
- Resumes processing

## Programming Considerations

This section describes things to consider when designing and coding FastExport job scripts.

### FastExport Configuration File

Use a FastExport configuration file to set the initial default values for the following operating parameters when FastExport is invoked:

- CHARSET
- ERRLOG
- BRIEF
- MAXSESS
- MINSSESS
- STATUS

- DATAENCRYPTION
- CONFIGERRORS

The values specified in the FastExport configuration file override the internal utility default values for these parameters.

The configuration file parameters themselves can be overridden by the commands in the FastExport job script, and by the corresponding run-time parameters, as shown in [Table 6 on page 25](#). The order of preference for these parameters, from highest to lowest, is:

- 1—FastExport script commands
- 2—Run-time parameters
- 3—FastExport configuration file specifications
- 4—FastExport default values

### Configuration File Name and Location

On network-attached systems, the FastExport configuration file must be named *fexpcfg.dat*

The file must be located in either:

- The directory from which FastExport is launched
- The directory specified in the FEXPLIB environment variable

On channel-attached systems, the DD statement for the FastExport configuration file must be labeled *FEXPCFG*.

### Configuration File Contents

The FastExport configuration file can have up to eight entries, one for each parameter:

```
CHARSET=character-set-name
ERRLOG=filename
BRIEF=on/off
MAXSESS=max-sessions
MINSESS=min-sessions
STATUS=ON/OFF
DATAENCRYPTION=ON/OFF
CONFIGERRORS=IGNORE/TERMINATE
```

where

- *character-set-name* is the character set specification for the FastExport job
- *filename* is the alternate file specification for FastExport error messages
- BRIEF *on/off* configures the reduced print output specification for the FastExport job
- *max-sessions* is the MAXSESS specification for the maximum number of FastExport sessions logged on to the Teradata Database
- *min-sessions* is the MINSESS specification for the minimum number of FastExport sessions required to run the job
- STATUS ON/OFF specifies the five-minute status reporting for the FastExport job
- DATAENCRYPTION ON/OFF specifies the data encryption option for the FastExport job
- IGNORE/TERMINATE configures the option for the FastExport configuration file error handling.

The FastExport configuration file can also have comment statements preceded by a number sign (#) character.

For a complete description of the parameter specifications listed here, see [Table 5 on page 23](#) and [Table 6 on page 25](#).

## Configuration File Processing

FastExport automatically checks for a configuration file each time the invocation command is entered. Upon locating a configuration file, the utility sets the defaults as specified, produces the appropriate output messages, and begins processing the FastExport job.

By default, any invalid configuration file entry or syntax error will abort the FastExport job immediately. The first invalid parameter will be reported, the rest of the configuration parameters will not be checked.

However, if CONFIGERRORS=IGNORE is specified in the configuration file, any subsequent configuration file problems will be reported but not affect the FastExport return code. FastExport will continue to process the next entry in the configuration file.

If CONFIGERRORS=TERMINATE is specified in the configuration file, any subsequent invalid configuration file entry will abort the FastExport job.

If there is no configuration file, the utility begins processing the FastExport job without an error indication. The configuration file is an optional feature of FastExport, and its absence is not considered to be an error condition.

## Generated MultiLoad Script File

When the MLSCRIPT option is specified in the EXPORT command, FastExport uses the functional parameters of the export task to generate a MultiLoad script file that can be used later to reload the export data back into the Teradata Database.

The following subsections provide an example of a generated MultiLoad script file and describe the changes which might need to be made before running the file, including the impact of OUTMOD routines and multiple select statements in FastExport job.

### Script File Example

```
/* Date of extract:          SUN JUN 27, 1999 */
/* Time of extract:         16:59:15          */

/* Total records extracted for select 1 = 2 */
/* Output record length for select 1 = 41 variable */

/* NOTE: THE SCRIPT BELOW MAY NEED TO BE MODIFIED BEFORE RUNNING. */

.LOGTABLE LOGTABLE165915;

.LOGON slugger/fexp_usr,fexp_usr;

.SET DBASE_TARGETTABLE TO 'fexp_usr';
.SET DBASE_WORKTABLE   TO 'fexp_usr';
.SET DBASE_ETTABLE     TO 'fexp_usr';
```

```

.SET DBASE_UVTABLE      TO 'fexp_usr';
.SET TARGETTABLE       TO 'TABLE165915';

.BEGIN IMPORT MLOAD
      TABLES &DBASE_TARGETTABLE..&TARGETTABLE
      WORKTABLES &DBASE_WORKTABLE..WT_&TARGETTABLE
      ERRORTABLES &DBASE_ETTABLE..ET_&TARGETTABLE
                &DBASE_UVTABLE..UV_&TARGETTABLE;

.LAYOUT DATAIN_LAYOUT;
.FIELD COL001 1 INTEGER;
.FIELD COL002 5 DATE;
.FIELD COL003 9 CHAR(5);
.FIELD COL004 14 VARCHAR(8);
.FIELD COL005 * BYTEINT;
.FIELD COL006 * SMALLINT;
.FIELD COL007 * DECIMAL(5,2);
.FIELD COL008 * BYTE(2);
.FIELD COL009 * VARBYTE(3);
.FIELD COL010 * FLOAT;

.DML LABEL INSERT_DML;
INSERT INTO &DBASE_TARGETTABLE..&TARGETTABLE (
  COL001 = :COL001
, COL002 = :COL002
, COL003 = :COL003
, COL004 = :COL004
, COL005 = :COL005
, COL006 = :COL006
, COL007 = :COL007
, COL008 = :COL008
, COL009 = :COL009
, COL010 = :COL010
);

.IMPORT INFILE dedtfm09.dat
      FORMAT FASTLOAD
      LAYOUT DATAIN_LAYOUT
      APPLY INSERT_DML;

.END MLOAD;

.LOGOFF &SYSRC;

/* End of script */

```

## Modifying the MultiLoad Script File

If necessary, modify the following specifications in the generated MultiLoad script file:

- Target table name
- Column names in the INSERT statement
- Field names in the LAYOUT statement
- Logon string
- Log table name

- Database of the following tables:
  - Target tables
  - Work tables
  - Error tables
  - Unique violation tables

Additionally:

- If the FastExport job uses an OUTMOD routine, the layout specifications in the generated MultiLoad script may need to be changed if the OUTMOD routine changes the record length
- If the FastExport job uses multiple SELECT statements, the response rows must all have the same structure. If the response rows have different structures, then the layout in the generated MultiLoad script will not work. In this case, to use the generated MultiLoad script file option:
  - Use a separate FastExport task for each SELECT statement in a FastExport job
  - In each corresponding EXPORT command, use the MLSCRIPT option with a different *fileid* specification

## FastExport Command Conventions

The following are command conventions to observe in FastExport job scripts.

### Conditional Expressions

Conditional expressions return a value of:

- 0 if the condition evaluates to FALSE
- 1 if the condition evaluates to TRUE

With the following exceptions, FastExport handles conditional expressions as described in the Teradata SQL reference documentation for the operating system environment:

- The LIKE operator is not supported in logical expressions that make up a conditional expression. (The NOT IN operator *is* supported.)
- The following elements are not supported in arithmetic expressions that make up logical expressions:
  - The exponential operator
  - Aggregate operators
  - Arithmetic functions

### Operators

Do not use words that are logical operators as keywords:

AND	IN	MOD
BETWEEN	IS	NE
EQ	LE	NOT
GE	LIKE	NULL
GT	LT	OR

## Reserved Words

Commands that are supported by FastExport do not use reserved words, except:

- Those that are operators
- Where a specific expression is allowed

Though there is no specific restriction against doing so, it is recommended that the following be avoided as variable names:

- FastExport command keywords
- Teradata SQL reserved words

## Variables

FastExport supports the following variables.

### Predefined System Variables

Table 7 lists the supported predefined system variables.

**Note:** System variables can only be referenced. They cannot be modified.

Table 7: System Variables

Variable Name	Description
&SYSDATE	Eight-character date in <i>yy/mm/dd</i> format
&SYSDATE4	Ten-character date in <i>yyyy/mm/dd</i> format
&SYSDAY	Three-character uppercase day of week specification: MON, TUE, WED, THU, FRI, SAT, or SUN
&SYSOS	Client operating system: <ul style="list-style-type: none"> <li>• For z/OS: VS1, z/OS, z/OS/SP, z/OS/ES</li> <li>• For z/VM: z/VM/SP, z/VM/XA SP, z/VM/HPO, z/VM/XA, z/VM/ESA</li> <li>• UNIX</li> <li>• Win32</li> </ul>
&SYSRC	Completion code of the last response from the Teradata Database
&SYSTIME	Eight-character time in <i>hh:mm:ss</i> format

Table 7: System Variables (continued)

Variable Name	Description
&SYSUSER	Client system dependent: <ul style="list-style-type: none"> <li>z/VM <i>userid</i></li> <li>z/OS batch <i>userid</i>. (z/OS batch returns <i>userid</i> only when a security package such as RACF, ACF2, or Top Secret has been installed).</li> </ul>

### Date and Time Variables

The four date and time variables reflect the time when FastExport begins execution:

- &SYSDAY
- &SYSDATE
- &SYSDATE4
- &SYSTIME

The original values are maintained after a FastExport restart operation.

**Note:** Do not reference the values in numeric operations since the values are all character data types.

### ANSI/SQL DateTime Specifications

ANSI/SQL DateTime specifications must be converted to fixed-length CHAR data types when specifying column/field names in the FIELD command.

For a description of the fixed-length CHAR representations for each DATE, TIME, TIMESTAMP, and INTERVAL data type specification, see [“Usage Notes” on page 88](#).

### Comments

[Table 8](#) describes the C language style comments supported by FastExport.

Table 8: C Language Comments

Comment Topic	Description
Beginning and Ending Delimiters	A comment begins with a slash asterisk (/*) character sequence and ends with an asterisk slash (*/) sequence. All intervening text is treated as a comment.
Comment Destinations	Comments are always written to the message destination, and they may or may not be sent to the Teradata Database.  Comments that are followed by a semicolon character are considered to be stand-alone comments:  <pre>/*Comment text*/; SELECT C1 FROM TABLE1;</pre> In this case, the comment is associated with the SELECT statement and is sent to the Teradata Database.



Table 8: C Language Comments (continued)

Comment Topic	Description
Invalid Within String or Character Literals	Comments cannot occur within string or character literals. A /* within a quoted string is not treated as the beginning of a comment.
Nested Comments	FastExport supports nested comments, but the Teradata Database does not.  Always delimit nested comments with a semicolon character.  If a semicolon is used to delimit a nested comment, it is taken as part of the current command or statement. If that happens to be a Teradata SQL statement, it will be sent to the Teradata Database, producing a syntax error.
Variable Substitution	Substitution of values for variable names continues within comments. Use two ampersand characters (&&) when the variable name is required.
Using Comments With Teradata SQL Statements	If a comment is used with a Teradata SQL statement, add a semicolon to the end of the comment if the comment should not be sent to the Teradata Database.  If a semicolon is not used, FastExport sends the comment to the Teradata Database along with the Teradata SQL statement.

## Character Set Specification

Teradata Database allows a character set to be established when invoking FastExport. For example, if a table or database names that have kanji double-byte characters or mixed single-byte and multibyte characters, the appropriate character set can be chosen.

[Table 9](#) lists the standard character sets supported by FastExport.

Table 9: Standard Character Sets Supported by FastExport

Name	Description	System Configuration
EBCDIC	Latin	Channel-attached
ASCII	Latin	Network-attached
HANGULEBCDIC933_1II	Korean	Channel-attached
HANGULKSC5601_2R4	Korean	Network-attached
KATAKANAEBDCIC	Japanese	Channel-attached
KANJIEBCDIC5026_0I	Japanese	Channel-attached
KANJIEBCDIC5035_0I	Japanese	Channel-attached
KANJIEUC_0U	Japanese	Network-attached
KANJISJIS_0S	Japanese	Network-attached
SCHEBCDIC935_2IJ	Simplified Chinese	Channel-attached

Table 9: Standard Character Sets Supported by FastExport (continued)

Name	Description	System Configuration
SCHGB2312_1T0	Simplified Chinese	Network-attached
TCHBIG5_1R0	Traditional Chinese	Network-attached
TCHEBCDIC937_3IB	Traditional Chinese	Channel-attached
UTF-8	Unicode character set	Network-attached
UTF-16	Unicode character set	Network-attached

### Site-Defined Character Sets

When the character sets defined are not appropriate for a site, define the character sets shown in [Table 10](#).

Table 10: Site-Defined Character Sets

Name	Description	System Configuration
SDKATAKANAEBDIC_4IF	Site-defined Japanese	Channel-Attached
SDKANJIEBCDIC5026_4IG	Site-defined Japanese	Channel-Attached
SDKANJIEBCDIC5035_4IH	Site-defined Japanese	Channel-Attached
SDKANJIEUC_1U3	Site-defined Japanese	Network-Attached
SDKANJISJIS_1S3	Site-defined Japanese	Network-Attached
SDSCHEBCDIC935_6IJ	Site-defined Simplified Chinese	Channel-attached
SDTCHEBCDIC937_7IB	Site-defined Traditional Chinese	Channel-attached
SDSCHGB2312_2T0	Site-defined Simplified Chinese	Network-Attached
SDTCHBIG5_3R0	Site-defined Traditional Chinese	Network-Attached
SDHANGULEBCDIC933_5II	Site-defined Korean	Channel-Attached
SDHANGULKSC5601_4R4	Site-defined Korean	Network-Attached

**Note:** For information about defining a character set appropriate for a site, see *International Character Set Support*.

### Rules for Using Chinese and Korean Character Sets

Observe the following rules when using Chinese and Korean character sets on channel-attached and network-attached platforms:

- Object Names  
Object names are limited to A-Z, a-z, 0-9, and special characters such as \$ and \_.
- Maximum String Length

The Teradata Database 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.

**Note:** For more information about Chinese or Korean character set restrictions for the Teradata Database, or for more information about alternate character sets, see *International Character Set Support*.

If Japanese language support is not required, specify EBCDIC or ASCII as the character set parameter.

## Unicode Character Sets

UTF-8 and UTF-16 are two of the standard ways of encoding Unicode character data. The UTF8 client character set supports UTF-8 encoding. Currently, Teradata Database supports UTF-8 characters that can consist of from one to three bytes. The UTF16 client character set supports UTF-16 encoding. Currently, the Teradata Database supports the Unicode 2.1 standard, where each defined character requires exactly 16 bits.

There are restrictions imposed by Teradata Database on using the UTF8 or UTF16 character set. For restriction details, see *International Character Set Support*.

### UTF8 Character Sets

FastExport supports UTF8 character set on network-attached platforms and IBM z/OS. When using UTF8 client character set on IBM z/OS, the job script must be in Teradata EBCDIC. FastExport translates commands in the job script from Teradata EBCDIC to UTF8 during the export.

Be sure to check the definition in *International Character Set Support* to determine the code points of any special characters required in the job script.

Different versions of EBCDIC do not always agree as to the placement of these characters. See Appendix E of *International Character Set Support* for details on mapping Teradata EBCDIC and Unicode.

### UTF16 Character Sets

FastExport supports UTF16 character set on network-attached platforms. In general, the command language and the job output should be the same as the client character set used by the job. However, for users' convenience and because of the special property of Unicode, the command language and the job output are not required to be the same as the client character set when using UTF16 character set. When using UTF16 character set, the job script and the job output can either be in UTF-8 or UTF-16 character set. This is provided by specifying runtime parameters "-i" and "-u" when the job is invoked.

For more information on runtime parameters "-i" and "-u", see parameters `-i scriptencoding` and `-u outputencoding` in [Table 6 on page 25](#).

[Table 11](#) describes four ways to either specify the character set or accept a default specification.

Table 11: Methods for Specifying Character Sets

Method	Description
Client System Specification	<p>Another way is to specify the character set for a client system before invoking FastExport by configuring the:</p> <ul style="list-style-type: none"> <li>• HSHSPB parameter for channel-attached z/VM and z/OS client systems</li> <li>• <i>dispb.dat</i> file for network-attached UNIX and Windows client systems</li> </ul> <p><b>Note:</b> The <i>character-set-name</i> specification used when to invoke FastExport always takes precedence over the current client system specification.</p>
FastExport Utility Default	<p>If there is no character set specification in DBC.Hosts, then FastExport defaults to:</p> <ul style="list-style-type: none"> <li>• EBCDIC for channel-attached z/VM and z/OS client systems</li> <li>• ASCII for network-attached UNIX client systems</li> </ul>
Run-time Parameter Specification	<p>The best way to specify the character set is with the character set run-time parameter when invoking FastExport, as described earlier in this chapter:</p> <ul style="list-style-type: none"> <li>• <code>CHARSET=<i>character-set-name</i></code> for channel-attached z/VM and z/OS client systems</li> <li>• <code>-c <i>character-set-name</i></code> for network-attached UNIX and Windows client systems</li> </ul> <p>For a list of valid character set names, see “<a href="#">Character Set Specification</a>” on page 41.</p>
Teradata Database Default	<p>If a <i>character-set-name</i> specification is not used when FastExport is invoked, and there is no character set specification for the client system, then the utility uses the default specification in the Teradata Database system table DBC.Hosts.</p> <p><b>Note:</b> If the DBC.Hosts table specification is relied upon for the default character set, make sure that the initial logon is in the default character set:</p> <ul style="list-style-type: none"> <li>• EBCDIC for channel-attached z/VM and z/OS client systems</li> <li>• ASCII for network-attached UNIX and Windows client systems</li> </ul>

## Using AXSMOD

When an AXSMOD is used, FastExport will pass the session character set as an attribute to the AXSMOD for its possible use (most AXSMODs will not make any use of this information). The attribute name will be `CHARSET_NAME` and it will be a variable length character string.

After FastExport passes the session character set to the AXSMOD successfully, FastExport will pass export widths information that pertains to the current session character set as an attribute to the AXSMOD for its possible use. The attribute name is `EXPORT_WIDTHS`. FastExport extracts the export widths information from the data parcel returned by the `HELP SESSION` command.

The export width information is passed as an array to the AXSMOD and is used by the AXSMOD to calculate the size in bytes of exported fixed-length character columns. This size depends not only on the number of characters in the data type (the  $n$  in CHAR( $n$ )), but also on the selected session character set, and the server character type (specified in the CHARACTER SET clause of the CREATE TABLE statement). Each structure passed in the array has information for one server character type. The export widths information structure is defined as the following:

```
typedef struct pmExpWidth
{
    pmUInt16 CharType;    /* Server character type code. */
    pmUInt16 ExpWidth;   /* Export width. */
    pmUInt16 ExpWidthAdj; /* Export width adjustment. */
} pmExpWidth_t;
```

For more information about export width rules, see *Utilities*.

## Multibyte Character Sets

Multibyte character sets impact the operation of certain FastExport commands, as well as object names in Teradata SQL statements, as shown in [Table 12](#).

Table 12: Commands Impacting Multibyte Character Sets

FastExport Command	Affected Elements	Impact
FIELD	Field name	The field name specified can have multibyte characters. In addition, it can be referenced in: <ul style="list-style-type: none"> <li>Other FIELD commands</li> <li>NULLIF and field concatenation expressions</li> <li>APPLY WHERE conditions in IMPORT commands</li> <li>Contain a NULLIF expression, which may use multibyte characters</li> </ul>
FILLER	Filler name	The name specified in a FILLER command can have multibyte characters.
LAYOUT	Layout name	The layout name can: <ul style="list-style-type: none"> <li>Have multibyte characters</li> <li>Be used in the LAYOUT clause of an IMPORT command</li> </ul>
	CONTINUEIF condition	The CONTINUEIF condition can specify multibyte character set character comparisons.
LOGON	User name and password	The user name and password can have multibyte characters.
LOGTABLE	Table and database names	The restart log table name and database name can have multibyte characters.

## Graphic Data Types

FastExport supports the following two-byte graphic data types in both the export data and the file containing the FastExport job script:

- GRAPHIC
- VARGRAPHIC
- LONG VARGRAPHIC

Use the `datadesc` parameter of the `FIELD` and `FILLER` commands to define graphic data types in the FastExport job script.

## Graphic Constants

FastExport supports two forms of graphic constants:

- The graphic literal or string constant, which is allowed in the KANJI EBCDIC character set on channel-attached z/VM and z/OS client systems. This type of constant must have an even number of bytes within the quoted string to represent double-byte characters.
- The hexadecimal representation of graphic data used on both:
  - Network-attached UNIX and Windows client systems
  - Channel-attached z/VM and z/OS client systems

For more information about graphic constants and hexadecimal representations of them, see information about Teradata SQL syntax and the lexicon in *SQL Fundamentals* for the operating system environment.

## Select Requests

A select request is one or more Teradata SQL `SELECT` statements that may be optionally preceded by a `LOCKING` modifier. The following information might apply for a FastExport job, depending on how select requests were created in that job.

### IMPORT Commands

If an `IMPORT` command is used, FastExport executes the select request once for each select data record, as specified by the `FIELD` commands. In this case, the response data for each execution of the select request is concatenated into the output data set.

If an `IMPORT` command defines input variables for the `WHERE` condition, the external names cited in the `WHERE` condition must correspond to the names in the `FIELD` commands.

**Note:** Each name in the `WHERE` condition is preceded by a colon character, while the names in the `FIELD` commands are not.

### Multiple SELECT Statements

If select request has multiple `SELECT` statements, the Teradata Database may execute them in parallel, but still returns the response data for the first statement first, then the response data for the second, and so on.

When you specify an OUTMOD routine to process the output records, the Teradata Database returns the FastExport job statement number with the response data for each SELECT statement. Otherwise, there is nothing returned to the output data set to delimit the records for one statement from those of the next.

### LOCKING Modifiers

If a LOCKING modifier is used, the specified lock remains in effect during execution of all statements within the request containing the modifier.

The Teradata Database:

- Implements all resource locks for the entire request before executing any of the statements in the request
- Maintains the locks until all of the response data for the request has been moved to spool tables

**Note:** The Teradata Database removes the resource locks before returning the data to the client system.

### Select Request Restrictions

FastExport select requests cannot:

- Specify a USING modifier. To submit data parameters as restraint parameters with a SELECT statement, they must be defined using a FastExport IMPORT command with supporting FIELD and FILLER commands.
- Access nondata tables, such as SELECT DATE or SELECT USER
- Be satisfied by a single AMP, such as a SELECT statement with a constraint containing an equality condition on the primary index or unique secondary index columns of a table

Other than these restrictions, the Teradata Database parses and processes SELECT statements from FastExport as it would from any other data access facility. For a complete description of the Teradata SQL SELECT statement, see the Teradata SQL reference documentation for the operating system environment.

## Restrictions and Limitations

Table 13 describes the FastExport restrictions and limitations on operational features and functions.

Table 13: FastExport Programming Restrictions and Limitations

Operational Feature/Function	Restriction/Limitation
Maximum file size	On UNIX MP-RAS operating system, the maximum file size that is supported by FastExport is 2 gigabytes. On Windows, Solaris SPARC, AIX, and HP-UX operating systems, there is no file size restriction.

Table 13: FastExport Programming Restrictions and Limitations (continued)

Operational Feature/Function	Restriction/Limitation
Concurrent Load Utility Tasks	<p>The maximum number of concurrent FastExport tasks that can run is variable; the limit can be controlled by the system administrator. MaxLoadTasks may be overridden if TASM is active.</p> <p><b>Note:</b> For the most up-to-date information on concurrent task limits, see the description of the MaxLoadTask parameter of the DBSControl utility in <i>Utilities Volume 1</i>. Additional information is also available in the <i>Teradata Dynamic Workload Manager User Guide</i>.</p> <p>If a FastExport job exceeds the recommended limits, the Teradata Database returns a 2633 error message indicating that too many loads are running, and the utility retries until:</p> <ul style="list-style-type: none"> <li>• It can execute the task</li> <li>• It reaches the TENACITY <i>hours</i> time limit specified by the BEGIN EXPORT command</li> </ul>
Exponential operators	Not allowed
Concatenation of data files	Not allowed
Expressions	Are evaluated from left to right, using the Teradata Database order of preference, but can be overridden by parentheses
Hexadecimal Form	FastExport does not accept and will not display object names specified in internal Teradata Database hexadecimal form.

## Termination Control Codes

When a FastExport job terminates, the utility returns a completion code to the client system:

- 00 = Normal completion
- 04 = Warning
- 08 = User error
- 12 = Severe internal error
- 16 = No message destination available

To avoid ambiguous or conflicting results, always use values greater than 20 when specifying a return code with the LOGOFF command.

## UNIX Signals

If running FastExport in a UNIX operating system, be aware of the UNIX signals used by FastExport. The FastExport UNIX signals in any program module or routine cannot be used with FastExport. Doing so causes an error in FastExport.

FastExport uses the following UNIX signals:

- SIGINT (interrupt signal)



- SIGQUIT (quit signal)
- SIGTERM (terminate signal)

**Note:** Signals are predefined messages sent between two UNIX processes to communicate the occurrence of unexpected external events, or exceptions. Aborting a FastExport session while FastExport is in the middle of processing a job is an example of an exception. In this scenario, FastExport uses the UNIX signals to trap the abort command, disconnect all sessions, do any necessary cleanup, and then terminate in an orderly manner.

## Using INMOD, OUTMOD, and Notify Exit Routines

The following sections provide information about how to use input modification (INMOD), output modification (OUTMOD), and notify exit routines.

### Overview

This section describes the different types of routines and when they might be used.

#### INMOD and OUTMOD Routines

The terms INMOD and OUTMOD are acronyms for *input modification* and *output modification* routines. These are user-written routines that FastExport and other load/export utilities can call to provide enhanced processing functions on:

- Input records before they are sent to the Teradata Database (INMOD routines)
- Output records before they are sent to the client system (OUTMOD routines)

Table 14 illustrates how FastExport supports both INMOD and OUTMOD routine calls in the FastExport job script.

Table 14: INMOD and OUTMOD Routines

FastExport Command	Specify	Write to Routine
IMPORT	INMOD	Read and preprocess input data values from files on the client system. These would then provide the USING data for a subsequent SELECT statement.
EXPORT	OUTMOD	Validate and preprocess export data records from the Teradata Database before writing them to files on the client system.

#### Notify Exit Routines

A notify exit routine specifies a predefined action to be performed whenever certain significant events occur during a FastExport job.

Notify exit routines are especially useful in operator-free environments where job scheduling relies heavily on automation to optimize system performance.

For example, by writing an exit in C (without using CLIV2) and using the NOTIFY\_EXIT option of the BEGIN EXPORT command, a routine can be provided to detect whether a FastExport job succeeds or fails, how many records are exported, the return code for a failed job, and so on.

## Programming Considerations for Using Routines

This section describes programming languages supported for each type of routine, as well as other related considerations.

### Programming Languages

FastExport is written in:

- SAS/C for channel-attached z/VM and z/OS client systems
- C for network-attached UNIX and Windows client systems

In all cases, INMOD, OUTMOD, and notify exit routines are dynamically loaded at run time, rather than link edited into the FastExport module.

INMOD, OUTMOD, and notify exit routines are written in the programming languages listed in [Table 15](#). The routines are dependent on the platform which runs FastExport.

Table 15: Languages Supported by Platform and Type of User-Developed Routine

Platform	INMOD Routines	OUTMOD Routines	Notify Exit Routines
z/VM, z/OS	Assembler, COBOL, PL/I, SAS/C	Assembler, COBOL, SAS/C	SAS/C
UNIX, Windows	C	C	C

**Note:** Although it is neither certified nor supported, INMOD and OUTMOD routines can be written in COBOL on network-attached client systems if the Micro Focus COBOL for UNIX compiler is used.

### Programming Structure

Programming structures for INMOD, OUTMOD, and notify exit routines differ, as reflected in the following sections.

#### ***INMOD Routines***

[Table 16](#) lists the programming language structures for communicating between FastExport and an INMOD routine.

Table 16: Programming Structure for INMOD Routines

INMOD Routine Language	Programming Structure
Assembler	<p><b>First parameter:</b>  RRECORD DSECT  RTNCODE DS F  RLENGTH DS F  RBODY DS CL32004</p> <p><b>Second parameter:</b>  IPARM DSECT  RSEQ DS F  PLEN DS H  PBODY DS CL100</p>
C	<p><b>First parameter:</b>  struct {      long Status;      long RecordLength;      char buffer[32004];  }</p> <p><b>Second parameter:</b>  struct {      long seqnum;      short parmlen;      char parm[80];  }</p>
COBOL	<p><b>First parameter:</b>  01 INMOD-RECORD.      03 RETURN-CODE PIC S9(9) COMP.      03 RECORD-LENGTH PIC 9(9) COMP.      03 RECORD-BODY PIC X(32004)</p> <p><b>Second parameter:</b>  01 PARM-STRUCT.      03 SEQ-NUM PIC 9(9) COMP.      03 PARM-LEN PIC 9(4) COMP.      03 PARM-BODY PIC X(80).</p>
PL/I	<p><b>First parameter:</b>  DCL 1 PARMLIST,      10 STATUS FIXED BINARY(31,0)      10 RLENGTH FIXED BINARY(31,0)      10 REC CHAR(32004)</p> <p><b>Second parameter:</b>  DCL 1 PARMLIST,      10 SEQNUM FIXED BINARY(31,0)      10 PLENGTH FIXED BINARY(15,0)      10 PBODY CHAR(80)</p>

In each structure, the records must be constructed so that the left-to-right order of the data field corresponds to the order of the field names specified in the FastExport LAYOUT and subsequent FIELD and FILLER commands.

### **OUTMOD Routines**

The structure for communicating between FastExport and an OUTMOD routine uses the standard C calling conventions for the following parameters:

- int \*code;
- int \*stmtnum;
- int \*InLen;
- char \*InBuf;
- int \*OutLen;
- char \*OutBuf;

### Notify Exit Routines

The structure for communicating between FastExport and a notify exit routine is a pointer to an FXNotifyExitParm structure, as shown in [“Sample Notify Exit Routine” on page 172](#).

### Routine Entry Points

[Table 17](#) shows the entry points for INMOD, OUTMOD, and notify exit routines.

Table 17: Entry Points for INMOD, OUTMOD, and Notify Exit Routines

Routine Language	Entry Point
SAS/C on z/OS and z/VM platforms,	_dynamn
C on UNIX and Windows platforms,	_dynamn (or BLKEXIT*) *Only for FDL-compatible INMODs compiled and linked with BLKEXIT as the entry point. When the FDL-compatible INMOD is used, 'USING("FDLINMOD")' must be specified in the .IMPORT statement.
COBOL and PL/I,	DYNAMN

### Working with Multiple Routines

For each export task associated with a SELECT statement, the FastExport job can specify:

- One INMOD routine with the IMPORT command
- One OUTMOD routine with the EXPORT command
- One notify exit routine with the BEGIN EXPORT command

A FastExport job can specify multiple export tasks, and each one can specify an INMOD routine, an OUTMOD routine, and a notify exit routine. (These specifications can be to the same or different routines.)

### Compiling and Linking Routines

The methods for compiling and linking routines vary with the operating system. The following sections describe the methods for z/VM, z/OS, UNIX, and Windows.

For sample programs and procedures that compile and link INMOD, OUTMOD, and notify exit routines for the operating system environment, see [Appendix C: “INMOD, OUTMOD and Notify Exit Routine Examples.”](#)

### **z/VM**

On channel-attached z/VM client systems, routines must be compiled and passed to CLINK with the following options:

- CLINK <filename>
- LKED
- LIBE
- DYNAMC
- NAME <modulename>

The resulting module, which can be loaded by SAS/C at run time, is placed in a load library called DYNAMC LOADLIB. (The first name must be DYNAMC because this is the only place that SAS/C looks for user load modules.)

Multiple load modules can exist in the local library as long as each module has a unique name.

### **z/OS**

The procedure on z/OS platforms is similar to the procedure on z/VM platforms, with one exception: user load modules can be located anywhere, as long as the location is identified by one of the DDNAME STEPLIB specifications in the JCL.

### **UNIX**

On network-attached UNIX client systems, INMOD, OUTMOD, and notify exit routines must:

- Be compiled with the MetaWare High C compiler
- Be linked into a shared object module
- Use an entry point named `_dynamn`

### **Windows**

On network-attached Windows client systems, INMOD, OUTMOD, and notify exit routines must:

- Be written in C
- Have a `dynamn` entry point that is a `__declspec`
- Be saved as a dynamic link library (DLL) file

### **Addressing Mode on z/VM and z/OS Systems**

On FastExport 07.00.00 and later, use either 31-bit or 24-bit addressing for INMOD, OUTMOD, and notify exit routines on channel-attached systems.

The 31-bit mode provides access to more memory, which enhances performance for FastExport jobs with a large number of sessions.

Use the following linkage parameters to specify the addressing mode when building INMOD, OUTMOD, and notify exit routines for z/VM and z/OS systems:

- For 31-bit addressing:  
`AMODE (31) RMODE (24)`

- For 24-bit addressing:

AMODE (24) RMODE (24)

## FastExport/INMOD Routine Interface

FastExport exchanges information with an INMOD routine by using the conventional parameter register to point to a parameter list of two 32-bit addresses.

The first 32-bit address points to a three-value INMOD interface parameter list consisting of status code, length, and body values. The second 32-bit address of the FastExport/INMOD interface points to a data structure containing sequence number and parameter list fields.

The following sections describe both sets of address pointers.

### Status Code

The status code pointer is a 32-bit signed binary value that carries information in both directions. [Table 18](#) explains the eight status code values of the FastExport-to-INMOD interface.

Table 18: FastExport-to-INMOD Status Codes

Value	Description
0	FastExport is calling for the first time and expects the INMOD routine to return a record. At this point, the INMOD routine should perform its initialization tasks before sending a data record to FastExport.
1	FastExport is calling, <i>not</i> for the first time, and expects the INMOD routine to return a record.
2	The client system has been restarted and the INMOD routine should reposition to the last checkpoint. FastExport is not expecting the INMOD routine to return a record. This is a one-time call, and FastExport does not issue a subsequent call with a status code value of zero. <b>Note:</b> If the client system restarts before the first checkpoint, entry code 1 is sent to reinitialize the job.
3	A checkpoint has been written and the INMOD routine should save the checkpoint position. FastExport does not expect the INMOD routine to return a record.
4	The Teradata Database has failed and the INMOD routine should reposition to the last checkpoint. FastLoad is not expecting the INMOD routine to return a record. This is a one-time call, and FastExport does not issue a subsequent call with a status code value of zero. If the database restarts before the first checkpoint, entry code 2 is sent for cleanup, and entry code 1 is sent to re-initialize the job.
5	The FastExport job has ended and the INMOD routine should perform any required cleanup tasks. <b>Note:</b> This condition applies only to network-attached client systems.

Table 18: FastExport-to-INMOD Status Codes (continued)

Value	Description
6	The INMOD routine should initialize and prepare to receive a record from FastExport.
7	The INMOD routine should receive a record from FastExport.

Table 19 reflects the INMOD-to-FastExport interface status code values (two).

Table 19: INMOD-to-FastExport Interface Status Codes

Status Code	Indication
0	The INMOD routine: <ul style="list-style-type: none"> <li>is returning a record as the Body value in response to a read call</li> <li>has successfully completed a nonread call</li> </ul>
Any nonzero value	The INMOD routine has encountered: <ul style="list-style-type: none"> <li>the end-of-file condition in response to a read call</li> <li>a processing error in response to a nonread call</li> </ul>

## Length

Length is a 32-bit signed binary value that specifies the length, in bytes, of the data record. The maximum record length is 62K, or 63,488 bytes.

The INMOD routine can use a Length value of zero to indicate an end-of-file condition.

## Body

Body is the area where the INMOD routine places the data record.

## Sequence Number

The sequence number field contains a 4-byte integer, which is the integer record counter portion of the source sequence number.

## Parameter List

The parameter list field consists of:

- VARCHAR specification
- Two-byte length specification, *m*
- The *m*-byte *parms* string, as parsed and presented by FastExport

**Caution:** To prevent data corruption, INMOD routines that cannot comply with these protocols should terminate if they encounter a restart code 2, 3, or 4. To support proper FastExport restart operations, INMOD routines must save and restore checkpoint information as described here. If the INMOD saves checkpoint information in some other manner, a subsequent restart/recovery operation could result in data loss or corruption.

## FastExport/OUTMOD Routine Interface

FastExport exchanges information with an OUTMOD routine by using the conventional parameter register to point to a six-value OUTMOD interface parameter list as described in the following sections.

### Entry Code

Entry code is one of six values that defines the reason for the call. [Table 20](#) describes the entry code values in 4-byte integer format. The values can be set as described in the table.

Table 20: FastExport-to-OUTMOD Interface Entry Codes

Code	Specifies
1	Initial Entry—Specifies the initial entry call that Fast Export makes before sending the first SELECT statement to the Teradata Database.
2	End of Response Entry—Specifies the end of response call that Fast Export makes after receiving the last row of export data from the Teradata Database.
3	Response Row Entry—Specifies a response row call that FastExport makes for each row of export data from the Teradata Database.
4	Checkpoint Entry—Specifies a checkpoint call that FastExport makes after processing the last response row for each SELECT statement.  This call signifies that the OUTMOD routine should capture checkpoint data to support a restart operation if the Teradata Database or client system fails.
5	Teradata Database Restart Entry—Specifies the first call that resumes processing after a Teradata Database restart.  If FastExport was writing to an output file when the Teradata Database failed, then the utility repositions that file before calling the OUTMOD routine. This call signifies that the OUTMOD routine should also reposition its data or files as needed and execute checkpoint procedures to resume processing.  If the database restarts before the first checkpoint, entry code 2 is sent for cleanup, and entry code 1 is sent to re-initialize the job.
6	Client Restart Entry—Specifies the first call that resumes processing after a client system restart.  If FastExport was writing to an output file when the client system failed, then the utility repositions that file before calling the OUTMOD routine. This call signifies that the OUTMOD routine should also reposition its data or files as needed and execute checkpoint procedures to resume processing.  If the client system restarts before the first checkpoint, entry code 1 is sent to re-initialize the job.

### Statement Number

Statement number specifies the *relative* statement number of the referenced SELECT statement—the first SELECT statement of the FastExport job being statement number 1.

This value is always 1 if the job has only one SELECT statement. The statement number is in 4-byte integer format.



## Input Data Length

Input data length specifies the length of the row data, in bytes, for a response row call.

If the OUTMOD routine requires FastExport to write the current response row data to the output file, then the OUTMOD routine should reset this value to 0 before returning to the utility.

The input data length is in 4-byte integer format.

## Input Data Record

Input data record specifies the buffer of the actual response row data from the Teradata Database.

## Output Data Length

Output data length specifies the length of the output data record. On entry to the OUTMOD routine, this parameter has a zero value and the output data record parameter points to an empty buffer. If the OUTMOD routine modifies an input data record, and the new length is:

- Less than or equal to the input length, the response row can be modified in place. In this case, the OUTMOD routine should set the input data length buffer to the new value before returning to FastExport.
- Greater than the input length, the output response row must be generated in the output data buffer. In this case, the OUTMOD routine should set the output data length buffer to the new value and reset the input data length buffer to zero before returning to FastExport.

The output data length is in 4-byte integer format.

**Caution:** To prevent data corruption, OUTMOD routines that cannot comply with these protocols should terminate if they encounter a restart code 4, 5, or 6. To support proper FastExport restart operations, OUTMOD routines must save and restore checkpoint information as described here. If the OUTMOD saves checkpoint information in some other manner, a subsequent restart/recovery operation could result in data loss or corruption.

## FastExport/Notify Exit Routine Interface

FastExport accumulates operational information about specific events that occur during a FastExport job. If the BEGIN EXPORT command includes a NOTIFY option with an EXIT specification, then, when the specific events occur, FastExport calls the named notify exit routine and passes to it:

- An event code to identify the event
- Specific information about the event

[Table 21](#) lists the event codes and descriptions of the data that FastExport passes to the notify exit routine for each event. (For a description of the events associated with each level of notification, see the description of the NOTIFY option in [“BEGIN EXPORT” on page 67.](#))

**Note:** To support future enhancements, always ensure that notify exit routines *ignore* invalid or undefined event codes, and that they *do not* cause FastExport to terminate abnormally.

Table 21: Events Passed to the Notify Exit Routine

Event Code	Event	Event Description	Data Passed to the Notify Exit Routine
0	Initialize	Successful processing of the NOTIFY option of the BEGIN EXPORT command	<ul style="list-style-type: none"> <li>Version ID length—4-byte unsigned integer</li> <li>Version ID string—32-character (maximum) array</li> <li>Utility ID—4-byte unsigned integer</li> <li>Utility name length—4-byte unsigned integer</li> <li>Utility name string—32-character (maximum) array</li> <li>User name length—4-byte unsigned integer</li> <li>User name string—64-character (maximum) array</li> <li>Optional string length—4-byte unsigned integer</li> <li>Optional string—80-character (maximum) array</li> </ul>
1	File or INMOD open	Successful processing of the IMPORT command that specifies the file or INMOD routine name	<ul style="list-style-type: none"> <li>File name length—4-byte unsigned integer</li> <li>File name—256-character (maximum) array</li> <li>Import number—4-byte unsigned integer</li> </ul>
9	Teradata Database restart	FastExport received a crash message from the Teradata Database or from the CLIV2	No data accompanies the Teradata Database restart event code
10	CLIV2 error	FastExport received a CLIV2 error	Error code—4-byte unsigned integer
11	Teradata Database error	FastExport received a Teradata Database error that will produce an exit code of 12	<p>Error code—4-byte unsigned integer</p> <p><b>Note:</b> Not all Teradata Database errors cause this event. For example, an Error 3807 that occurs while trying to drop or create a table does not terminate FastExport.</p>
12	Exit	FastExport is terminating	Exit code—4-byte unsigned integer
31	Export begin	FastExport is about to begin the export task by opening the export file	No data accompanies the export begin event code
32	Request submit begin	FastExport is about to submit the SELECT request to the Teradata Database	<ul style="list-style-type: none"> <li>Request length—4-byte unsigned integer</li> <li>Request text—32,000-character (maximum) array</li> </ul>
33	Request submit end	FastExport has received the response to the SELECT request	<ul style="list-style-type: none"> <li>Statement count—4-byte unsigned integer</li> <li>Block count—4-byte unsigned integer</li> </ul>
34	Request fetch begin	FastExport is about to fetch the results of the SELECT request	No data accompanies the request fetch begin event code
35	File or OUTMOD open	FastExport is about to open an output or OUTMOD routine file	<ul style="list-style-type: none"> <li>File name length—4-byte unsigned integer</li> <li>File name—256-character (maximum) array</li> </ul>
36	Statement fetch begin	FastExport is about to fetch the current statement in a request	<ul style="list-style-type: none"> <li>Statement number—4-byte unsigned integer</li> <li>Block count—4-byte unsigned integer</li> </ul>
37	Statement fetch end	FastExport has fetched all of the records for the current statement	Record count—4-byte unsigned integer

Table 21: Events Passed to the Notify Exit Routine (continued)

Event Code	Event	Event Description	Data Passed to the Notify Exit Routine
38	Request fetch end	FastExport has fetched all of the records for the current request	<ul style="list-style-type: none"> <li>Records exported—4-byte unsigned integer</li> <li>Records rejected—4-byte unsigned integer</li> </ul>
39	Export end	FastExport has completed the export operation and displayed the number of exported records	<ul style="list-style-type: none"> <li>Records exported—4-byte unsigned integer</li> <li>Records rejected—4-byte unsigned integer</li> </ul>

## Writing a FastExport Job Script

This section describes the contents of a FastExport job script and explains how to write one.

### Definition

A FastExport job script, or program, is a set of FastExport commands and Teradata SQL statements that select and export data from the Teradata Database.

Each FastExport job includes a number of support commands that establish and maintain the FastExport support environment, and a number of task commands that perform the actual database select and export operations.

---

### To write a FastExport job script

A complete FastExport job includes:

- Invoking FastExport
  - Logging on to the Teradata Database and establishing the FastExport support environment
  - Specifying the FastExport tasks
  - Logging off from the Teradata Database and terminating FastExport
- 1 Invoke FastExport, specifying desired run-time options:
    - Normal or abbreviated (brief) printout
    - Character set
    - Session limits
    - Alternate error message file
    - Standard input file/device
    - Standard output file/device
    - Alternate run file
  - 2 Establish the FastExport support environment as indicated in [Table 22](#).

Table 22: Commands for Establishing FastExport Support Environment

Command	Description
<a href="#">LOGTABLE</a>	Specify the restart log table that maintains checkpoint information for the FastExport job.
<a href="#">DATEFORM</a>	Specify the form of the DATE data type specifications for the FastExport job.
<a href="#">LOGON</a>	Establish a session with the Teradata Database.
<a href="#">RUN FILE</a>	Transfer processing control to an alternate run file.
<a href="#">ROUTE MESSAGES</a>	Specify an alternate destination for FastExport output messages.
<a href="#">SYSTEM</a>	Submit an operating system command to the client system.
Teradata SQL statements	Submit a supported Teradata SQL statement to the Teradata Database.

At a minimum, this portion of the FastExport job must include:

- A LOGTABLE command to specify the restart log table
- A LOGON command to provide a logon string that is used to connect all Teradata SQL and FastExport sessions with the Teradata Database

State these commands in any order, but both commands must appear before any task commands in the FastExport job script.

3 Specify the FastExport task as indicated in [Table 23](#).

Table 23: Commands for Specifying the FastExport Task

Command	Description
<a href="#">BEGIN EXPORT</a>	Signify the beginning of an export task specification.
<a href="#">LAYOUT</a>	Specify, in conjunction with immediately following FIELD and FILLER commands, the layout of the input data records on the client system. (The layout specification will be referenced in a subsequent IMPORT command.)
<a href="#">FIELD</a>	Specify a field of the input record to be used to supply values for variables in the SELECT statement.
<a href="#">FILLER</a>	Describe a field of the input data record that is not referenced, but used only to indicate the relative position of the following FIELD data.
<a href="#">EXPORT</a>	Specify the destination file and format specifications for export data retrieved from the Teradata Database.
<a href="#">IMPORT</a>	Specify the client file that provides the USING values for the SELECT statement.
<a href="#">SELECT</a>	Specify the row data to be exported from the Teradata Database.
<a href="#">END EXPORT</a>	Signify the end of the FastExport task and initiate the task processing.

At a minimum, this portion of the FastExport job must include:

- A BEGIN EXPORT command
  - A select request
  - An EXPORT command
  - An END EXPORT command
- 4 To specify another FastExport task:
- Use the support commands listed in Step 2 to modify the FastExport support environment for the next task
  - Use the task commands listed in Step 3 to specify the next task

Repeat these steps for each task in the FastExport job.

- 5 Use the LOGOFF command to disconnect all active sessions with the Teradata Database and terminate FastExport on the client system.

## Using Checkpoints in a Single Export Job

FastExport has a built-in feature for saving the position information for imported and exported data sources. This feature is useful with SELECT statements that retrieve a very large answer set, preventing FastExport from re-exporting data if a system restart occurs.

For example, assume a table of 5,000,000 rows exists, and the following single SELECT statement will be used to export the data to a file:

```
.EXPORT OUTFILE DATAFILE;
SEL * FROM DATA_TABLE;
.END EXPORT;
```

If a system restart occurs while this statement is processing, the entire export process must start over from the beginning.

Avoid this type of scenario by breaking the single SELECT statement into multiple statements using WHERE clauses and storing boundary values in an INFILE.

Break up the preceding single SELECT statement similar to the following example:

```
SEL * FROM DATA_TABLE WHERE FIELD1 BETWEEN          1 AND 1000000;
SEL * FROM DATA_TABLE WHERE FIELD1 BETWEEN 1000001 AND 2000000;
SEL * FROM DATA_TABLE WHERE FIELD1 BETWEEN 2000001 AND 3000000;
SEL * FROM DATA_TABLE WHERE FIELD1 BETWEEN 3000001 AND 4000000;
SEL * FROM DATA_TABLE WHERE FIELD1 BETWEEN 4000001 AND 5000000;
```

Place all boundary values in INFILE as follows:

```
-----
FIRST FIELD SECOND FIELD
-----
FIRST RECORD          1          1000000
SECOND RECORD        1000001        2000000
THIRD RECORD         2000001        3000000
FOURTH RECORD        3000001        4000000
FIFTH RECORD         4000001        5000000
-----
```

Then, as FastExport executes the multiple statements in the second example, they are sent one at a time. In addition, the position for the import and export data sources is saved.

If a system restart occurs while one of the statements is processing, FastExport does not restart the entire export process from the beginning as it does when a single SELECT statement is used. Instead, FastExport restarts the export process only from the statement that failed.

The following sample script uses the data from the INFILE for the WHERE clause and creates the multiple statements shown in the second example.

**Note:** This script can be used with FastExport running on all operating systems.

```
.LOGTABLE LOG_TBL;          /* define restart log          */
.LOGON TDPR/username, password; /* DBC logon string          */
.BEGIN EXPORT ;             /* specify export function     */
.LAYOUT BOUNDRIES;         /* define the input data values */
.FIELD FROMVALUE * INTEGER; /* for SELECT constraint clause */
.FIELD TOVALUE * INTEGER;

.IMPORT INFILE INDATAFILE   /* define the file that contains */
      LAYOUT BOUNDRIES;     /* the input data values        */

.EXPORT OUTFILE OUTPUTFILE ; /* identify the destination source*/
/* for exported data          */
SEL * FROM DATA_TABLE      /* provide the SQL SELECT       */
/* statement                   */
WHERE COLUMN1 BETWEEN      /* with values provided by the  */
/* IMPORT command              */
:FROMVALUE AND :TOVALUE ;

.END EXPORT;                /* terminate the export operation */
.LOGOFF;                    /* disconnect from the DBS      */
```

# FastExport Commands

---

This chapter describes the FastExport commands and Teradata SQL statements which can be executed from the FastExport utility.

Experienced FastExport users can also refer to the simplified command descriptions in the FastExport 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.

## Syntax Notes

Each FastExport command must:

- Begin on a new line
- Start with a period (.) character
- End with a semicolon (;) character

Each command can continue for as many lines as necessary, as long as it satisfies the beginning and ending requirements.

## Object Name Restrictions

The following Syntax rules apply to object names:

- A semicolon cannot be used in an object name because a semicolon is used to designate the end of a Teradata FastExport command.
- 30 bytes cannot be used as the quoted object name length.
- Dictionary (UDD) object names cannot be used.

See [Appendix A: “How to Read Syntax Diagrams”](#) for more information about how to read the syntax diagrams used in this book.

## Geospatial Data Restrictions

The following rules apply to Geospatial data:

- FastExport does not support Geospatial data represented by LOBs.
- FastExport does not support geospatial data beyond 64000.

# ACCEPT

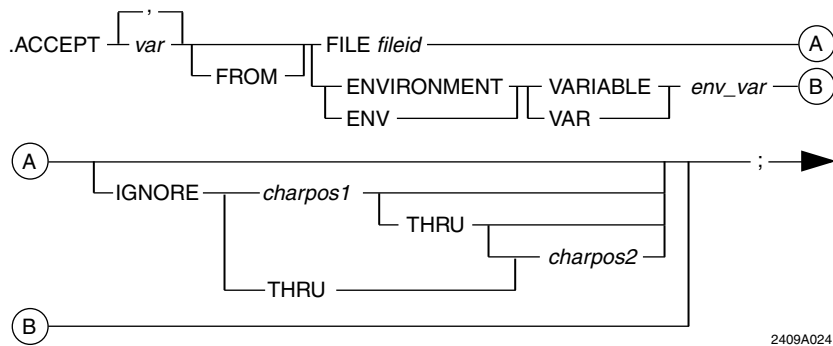
## Purpose

The ACCEPT command sets FastExport utility variables to the value of a specified:

- External data source and valid character fields
- Internal environment variable

The ACCEPT command is a valid command preceding LOGON and LOGTABLE commands.

## Syntax



where

Syntax Element	Description
<i>charpos1</i> and <i>charpos2</i>	Start and end character positions of a field in each input record that contains extraneous information For example: <ul style="list-style-type: none"> <li>• Use <i>charpos1</i> to ignore only the single specified character</li> <li>• Use <i>charpos1</i>THRU, to ignore all characters from <i>charpos1</i> through the end of the record</li> <li>• Use THRU <i>charpos2</i> to ignore all characters from the beginning of the record through <i>charpos2</i></li> <li>• Use <i>charpos1</i> THRU <i>charpos2</i> to ignore all characters from <i>charpos1</i> through <i>charpos2</i></li> </ul>
<i>env_var</i>	Environment variable that provides the value for the specified utility variables ( <i>var</i> )



Syntax Element	Description
<i>fileid</i>	<p>Data source of the external system</p> <p>The external system DD (or similar) statement specifies a file:</p> <ul style="list-style-type: none"> <li>In z/OS, <i>fileid</i> is a DDNAME. (See the “z/OS <i>fileid</i> Usage Rules” topic in the “Usage Notes” subsection.)</li> <li>In UNIX and Windows, <i>fileid</i> is the path name for a file. If the path name has embedded white space characters, they must enclose the entire path name in single or double quotes.</li> <li>In z/VM, <i>fileid</i> is a FILEDEF name</li> </ul>
<i>var</i>	<p>Name of the FastExport utility variable that is to be set with the value accepted from the designated source</p> <p>Character string values appear as quoted strings in the data file.</p>

## Usage Notes

Table 24 describes the things to consider when using the ACCEPT command.

Table 24: ACCEPT Command Usage Notes

Topic	Usage Notes
Specifying the System Console/ Standard Input Device	<p>The asterisk (*) character can be used as the <i>fileid</i> specification for the system console/standard input (stdin) device.</p> <p>The system console is the:</p> <ul style="list-style-type: none"> <li>Keyboard in interactive mode</li> <li>Standard input device in batch mode</li> </ul> <p>For more information about the keyboard and standard input devices, see “<a href="#">File Requirements</a>” on page 21.</p>
z/OS <i>fileid</i> Usage Rules	<p>If a DDNAME is specified, FastExport reads data records from the specified source.</p> <p>A DDNAME must obey the same construction rules as Teradata SQL column names, except that:</p> <ul style="list-style-type: none"> <li>The “at” character (@) is allowed as an alphabetic character</li> <li>The underscore character (_) is not allowed</li> </ul> <p>The DDNAME must obey the applicable rules of the external system. If the DDNAME represents a data source on magnetic tape, the tape may be either labeled or unlabeled, as supported by the operating system.</p>
Source File Record Restriction	<p>A single record, row, or input line is accepted from the designated source. Always make sure that there is only one record in the file from which the ACCEPT command is getting the variables.</p>
Coding Multiple Variables	<p>When multiple variables are coded, each is sequentially assigned input text up to the first space character encountered that is not within a quoted string.</p>

Table 24: ACCEPT Command Usage Notes (continued)

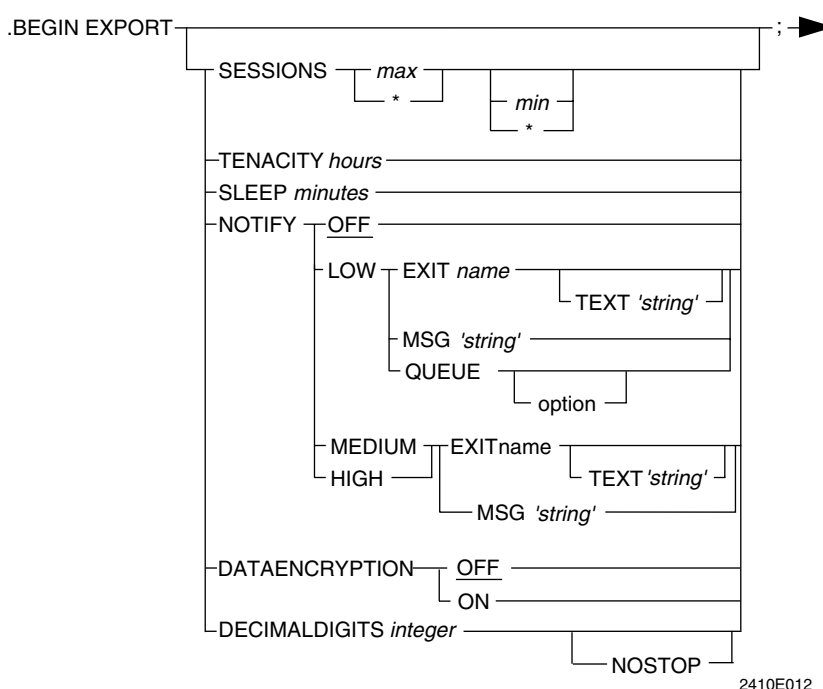
Topic	Usage Notes
Delimiting Input Text	<p>Input text for <i>numeric values</i> must be delimited only by space characters or record boundaries.</p> <p>Input text for <i>character strings</i> must be enclosed in apostrophes. For example, the data record provided to satisfy the following ACCEPT command should include two fields:</p> <pre>.Accept age, name from file info;</pre> <p>The following example shows two sample data records, where the first is correct but the second is not:</p> <pre>32 'Tom' 32 Tom</pre>
Number of Variables	<p>When the number of variables listed is greater than the number of responses available, unused variables remain undefined (null). If there are not enough variables to hold all responses, FastExport issues a warning message.</p>

# BEGIN EXPORT

## Purpose

The BEGIN EXPORT command signifies the beginning of an export task and sets the specifications for the task sessions with the Teradata Database.

## Syntax



where

Syntax Element	Description
SESSIONS...	<p>Maximum number of FastExport sessions that is logged on when a LOGON command is entered and, optionally, the minimum number of sessions required to run the job</p> <ul style="list-style-type: none"> <li>• The <i>max</i> value specifies the maximum number of sessions to log on. The <i>max</i> specification must be greater than zero. If a SESSIONS <i>max</i> value is specified that is larger than the number of available AMPs, FastExport limits the sessions to one per working AMP. The default maximum, if the SESSIONS option is not used, is 4.</li> <li>• The <i>min</i> value specifies the minimum number of sessions required for the job to continue. The <i>min</i> specification must be greater than zero. The default minimum, if the SESSIONS option is not used or a <i>min</i> value is specified, is 1.</li> <li>• The * value specifies the maximum and minimum number of sessions. Using the asterisk character as the <i>max</i> specification logs on for the maximum number of sessions—one for each AMP. Using the asterisk character as the <i>min</i> specification logs on for at least one session, but always less than or equal to the <i>max</i> specification.</li> </ul> <p>For more information about setting number of sessions, see <a href="#">“Usage Notes” on page 72</a>.</p>
TENACITY <i>hours</i>	<p>Number of hours that FastExport tries to log on to the Teradata Database</p> <p>When FastExport tries to log on for a new task, and the Teradata Database indicates that the maximum number of utility import/export sessions are already running, FastExport:</p> <ol style="list-style-type: none"> <li>1 Waits for six minutes, by default, or for the amount of time specified by the SLEEP option.</li> <li>2 Then it tries to log on to the Teradata Database again.</li> </ol> <p>FastExport repeats this process until it has either logged on for the required number of sessions or exceeded the TENACITY <i>hours</i> time period.</p> <p>The default value is 4.</p> <p>For more information about the maximum number of load utility tasks that can run, see <a href="#">“Concurrent Load Utility Tasks” on page 48</a>.</p>
SLEEP <i>minutes</i>	<p>Number of minutes that FastExport waits between logon attempts</p> <p>Default value is 6.</p> <p>FastExport uses the SLEEP specification in conjunction with the TENACITY specification.</p>

Syntax Element	Description
NOTIFY...	<p>FastExport implementation of the notify user exit option:</p> <ul style="list-style-type: none"> <li>• NOTIFY OFF suppresses the notify user exit option.</li> <li>• NOTIFY LOW enables the notify user exit option for those events signified by “Yes” in the Low Notification Level column of <a href="#">Table 25</a>.</li> <li>• NOTIFY MEDIUM enables the notify user exit option for those events signified by “Yes” in the Medium Notification Level column of <a href="#">Table 25</a>.</li> <li>• NOTIFY HIGH enables the notify user exit option for those events signified by “Yes” in the High Notification Level column of <a href="#">Table 25</a>.</li> </ul>
EXIT <i>name</i>	<p>User-defined exit where <i>name</i> is the name of a user-supplied library with a member name of <code>_dynamn</code></p> <p>The exit must be written in C, or in a programming language with a run-time environment that is compatible with C.</p> <p>For an example, see “<a href="#">Sample Notify Exit Routine</a>” on page 172.</p> <p><b>Note:</b> On some versions of UNIX, <code>./</code> prefix characters may have to be added to the EXIT <i>name</i> specification if the module is in the current directory.</p>
TEXT ' <i>string</i> '	<p>A user-supplied string of up to 80 characters that FastExport passes to the named user exit routine</p> <p>The <i>string</i> specification must be enclosed in single quote characters (<code>'</code>).</p>
MSG ' <i>string</i> '	<p>A user-supplied string of up to 16 characters that FastExport logs on to:</p> <ul style="list-style-type: none"> <li>• The operator’s console (channel-attached z/VM and z/OS client systems)</li> <li>• The system log (network-attached UNIX and Windows client systems)</li> </ul> <p>The <i>string</i> specification must be enclosed in single quote characters (<code>'</code>).</p>

Syntax Element	Description
<p>QUEUE <i>option</i></p>	<p>Queue management option on channel-attached z/OS client systems</p> <p><b>Note:</b> This option is available <i>only</i> on z/OS, and <i>only</i> for tasks with a low notification specification.</p> <p>This option invokes an ENQ when the BEGIN EXPORT command is processed, followed by a DEQ when the significant event occurs.</p> <p>The option specification is one of the following:</p> <p><b>RNAME</b></p> <p>A parameter containing a quoted string of up to 255 characters.</p> <p>The default is TDUSER.</p> <p><b>SCOPE</b></p> <p>A parameter that is one of the following:</p> <p>JOB—Specifies that the QUEUE is local to the job, including all job steps.</p> <p>SYSTEM—Specifies that the QUEUE is global to the computer running it.</p> <p>SYSTEMS—Specifies that the QUEUE is global to all computers in the complex.</p> <p>The default is SYSTEMS.</p> <p><b>NOBLOCK</b></p> <p>A parameter specifying that if the ENQ blocks for any reason, it must return an error instead. This is a fatal error for the job.</p> <p>The default, an implied BLOCK (there is no BLOCK keyword), means that the ENQ will wait for the QUEUE.</p>
<p>DATAENCRYPTION</p>	<p>Keyword that enables data encryption for the FastExport job</p> <p>Valid options are:</p> <ul style="list-style-type: none"> <li>• ON = All the requests between BEGIN EXPORT and END EXPORT commands will be encrypted.</li> <li>• OFF = The requests between BEGIN EXPORT and END EXPORT commands will not be encrypted. This is the default.</li> </ul> <p>This option will apply only to the requests between BEGIN EXPORT and END EXPORT commands.</p> <p>Using this option overwrites the data encryption settings specified by both the run-time parameters and in the <i>fexpcfg.dat</i> configuration file.</p>
<p>DECIMALDIGITS</p>	<p>A user-supplied maximum number of digits in the DECIMAL data type that can be exported. Starting from V2R6.2, the maximum number of digits in the DECIMAL data type increased from 18 to 38. Note that if a user doesn't set the limit, the default maximum number of digits is 18.</p> <p>When the client is a mainframe, the user can set the limit to 31 to request automatic CAST to avoid n&gt;31 results.</p> <p>Using this option overwrites the max_decimal_returned value specified in the <i>clispb.dat</i> file for network-attached systems or the HSHSPB parameter for channel-attached systems.</p>

Syntax Element	Description
NOSTOP	<p>If the NOSTOP option <i>is specified</i> and Teradata or CLIV2 does not support Large Decimal, if the user specifies a valid value for the decimaldigits parameter, FastExport does the following:</p> <ul style="list-style-type: none"> <li>• Displays a message that Teradata Database or Teradata CLIV2 does not support Large Decimal</li> <li>• Displays a warning that the decimaldigits setting is ignored</li> <li>• Continues with the Teradata FastExport job</li> <li>• Exits with an exit code of 4, unless there is another error with a higher exit code</li> </ul> <p>If the NOSTOP option <i>is not specified</i> and Teradata or CLIV2 does not support Large Decimal, if the user specifies a valid value for the decimaldigits parameter, FastExport maintains the current behavior and does the following:</p> <ul style="list-style-type: none"> <li>• Displays a message that Teradata Database or Teradata CLIV2 does not support Large Decimal</li> <li>• Terminates the FastExport job</li> <li>• Exits with an exit code of 8</li> </ul> <p><b>Note:</b> If the user specifies a valid value for the max_decimal_returned parameter in clispb.dat, FastExport maintains the current behavior, regardless of NOSTOP option.</p>

Table 25 lists the events which create notifications.

Table 25: Events That Create Notifications

Event	Notification Level			Signifies
	Low	Medium	High	
Initialize	Yes	Yes	Yes	Successful processing of the BEGIN EXPORT command
File or INMOD open	No	No	Yes	Successful processing of the IMPORT command
Teradata Database Restart	No	Yes	Yes	A crash error from the Teradata Database or the CLIV2
CLIV2 error	Yes	Yes	Yes	A CLIV2 error
Teradata Database error	Yes	Yes	Yes	A Teradata Database error that terminates FastLoad
Exit	Yes	Yes	Yes	FastExport is terminating
Export begin	No	Yes	Yes	Opening the export file
Request submit begin	No	Yes	Yes	Submitting the SELECT request
Request submit end	No	Yes	Yes	Received SELECT request response

Table 25: Events That Create Notifications (continued)

Event	Notification Level			Signifies
	Low	Medium	High	
Request fetch begin	No	Yes	Yes	Fetching SELECT request results
File or OUTMOD open	No	No	Yes	Opening output file or OUTMOD
Statement fetch begin	No	No	Yes	Fetching current statement
Statement fetch end	No	No	Yes	Last record fetched for current statement
Request fetch end	No	Yes	Yes	Last record fetched for current request
Export end	No	Yes	Yes	Export task completed

## Usage Notes

[Table 26](#) describes the things to consider when using the BEGIN EXPORT command.

Table 26: BEGIN EXPORT Usage Notes

Topic	Usage Notes
Command Placement and Frequency	<p>The BEGIN EXPORT command must be the first command in a group of FastExport utility commands that specify an export task.</p> <p>Multiple BEGIN EXPORT commands can be used in a FastExport job script, but each export task specification must begin with a BEGIN EXPORT command and end with an END EXPORT command.</p>



Table 26: BEGIN EXPORT Usage Notes (continued)

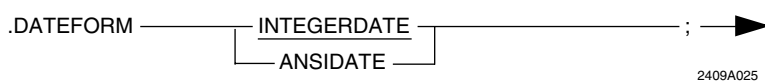
Topic	Usage Notes
Sessions Limit Specification	<p>The number of sessions that you should specify depends on the connections to the Teradata Database and the amount of data to be returned.</p> <p>In addition to the sessions that are used for the export task, FastExport uses two additional sessions to:</p> <ul style="list-style-type: none"> <li>• Maintain the restart log table</li> <li>• Submit Teradata SQL statements</li> </ul> <p>There is no general method to determine the optimal number of sessions, because it is dependent on several factors, including, but not limited, to:</p> <ul style="list-style-type: none"> <li>• Teradata Database performance and workload</li> <li>• Client platform type, performance, and workload</li> <li>• Channel performance, for channel-attached systems</li> <li>• Network topology and performance, for network-attached systems</li> <li>• Volume of data to be processed by the application</li> </ul> <p>When specifying the session limit, always consider the load that the export task is placing on the channel or network connection.</p> <p>For example, four sessions on a channel-attached system, each on a different interface processor (IFP) on a channel, and all concurrently returning data can saturate a single channel.</p> <p>In such a case, define the maximum number of sessions as four times the number of channels that are controlled by the Teradata Director Program (TDP) that connects the sessions.</p> <p>Using too few sessions is likely to unnecessarily limit throughput. On the other hand, using too many sessions can increase session management overhead (and also reduce the number of sessions available to any other applications) and may, in some circumstances, degrade throughput.</p> <p>If the minimum number of FastExport sessions are not logged, FastExport will terminate.</p> <p>Regardless of the size of the Teradata Database configuration, for large repetitive production applications, it will usually be appropriate to experiment with several different session configurations to determine the best trade-off between resource utilization and throughput performance.</p> <p>For larger Teradata Database configurations, it is appropriate to establish an installation default for the maximum number of sessions that is greater than four sessions, but less than one session per AMP. This can be done using an installation configuration file (see <a href="#">“FastExport Configuration File” on page 34</a>) or a standard run-time parameter (see <a href="#">“Run-time Parameters” on page 23</a>).</p> <p>An installation default for number of sessions, if specified in the configuration file, can be overridden in individual FastExport job scripts, when necessary.</p> <p>On large to very large Teradata Database configurations, the limit of one session per AMP when * is specified may be inappropriately large.</p>

# DATEFORM

## Purpose

The **DATEFORM** command specifies the form of the DATE data type specifications for the FastExport job.

## Syntax



where

Syntax Element	Description
ANSIDATE	Keyword that specifies ANSI fixed-length CHAR(10) DATE data types for the FastExport job
INTEGERDATE	Keyword that specifies integer DATE data types for the FastExport job This is the default specification for FastExport jobs if a DATEFORM command is not entered.

## Usage Notes

[Table 27](#) describes the things to consider when using the DATEFORM command.

Table 27: DATEFORM Command Usage Notes

Topic	Usage Notes
Command Frequency and Placement	Only one DATEFORM command can be used. The command must be entered before the LOGON command.
Data Type Conversions	When the ANSIDATE specification is used, ANSI/SQL DateTime data types must be converted to fixed-length CHAR data types when specifying the column/field names in the FIELD command.  For each DATE, TIME, TIMESTAMP, and INTERVAL data type specification, see the “Usage Notes” subsection of the FIELD command description for a description of the fixed-length CHAR representations.

# DISPLAY

## Purpose

The DISPLAY command writes messages to a specified destination.

## Syntax

```
.DISPLAY 'text' [TO] FILE fileid ;
```

2409A027

where

Syntax Element	Description
'text'	Text to be written to the specified output destination
fileid	Data source of the external system The external system DD (or similar) statement specifies a file: <ul style="list-style-type: none"> <li>In z/OS, <i>fileid</i> is a DDName. (See the “z/OS <i>fileid</i> Usage Rules” topic in the “Usage Notes” subsection.)</li> <li>In UNIX and Windows, <i>fileid</i> is the path name for a file</li> <li>In z/VM, <i>fileid</i> is a FILEDEF name</li> </ul>

## Usage Notes

[Table 28](#) describes the things to consider when using the DISPLAY command.

Table 28: DISPLAY Command Usage Notes

Topic	Usage Notes
Conflicting Write Operations on Network-attached Systems	On network-attached client systems, if the same file is specified to redirect stdout as the file in a DISPLAY command, the results may be incomplete due to conflicting write operations to the same file.
Displaying Apostrophes in the Text String	To display an apostrophe within the text string, use two consecutive apostrophes (single quotes) to distinguish it from both the single quotes enclosing the string and a regular double-quote character.

Table 28: DISPLAY Command Usage Notes (continued)

Topic	Usage Notes
Specifying the System Console/Standard Output Device	<p>The asterisk (*) character can be used as the <i>fileid</i> specification to direct the display messages to the system console/standard output (stdout) device.</p> <p>The system console is the:</p> <ul style="list-style-type: none"> <li>• Display screen in interactive mode</li> <li>• Standard output device in batch mode</li> </ul> <p>For more information about the display screen and standard output devices, see <a href="#">“File Requirements” on page 21</a>.</p>
Utility Variables	<p>Utility variables are replaced by their values before text is displayed. This is done by preceding the variable name with an ampersand (&amp;) character.</p> <p>To display the name of a utility variable, code two ampersand characters instead of one.</p>
<i>z/OS fileid</i> Usage Rules	<p>A DDNAME must obey the same construction rules as Teradata SQL column names except that:</p> <ul style="list-style-type: none"> <li>• The "at" character (@) is allowed as an alphabetic character.</li> <li>• The underscore character (_) is not allowed.</li> </ul> <p>The DDNAME must obey the applicable rules of the external system.</p> <p>If the DDNAME represents a data source on magnetic tape, the tape may be either labeled or nonlabeled, as supported by the operating system.</p>

# END EXPORT

## Purpose

The END EXPORT command signifies the end of an export task and initiates processing by the Teradata Database.

## Syntax

```
.END EXPORT _____ ;
```

2410A016

## Usage Notes

[Table 29](#) describes the things to consider when using the END EXPORT command.

Table 29: END EXPORT Command Usage Notes

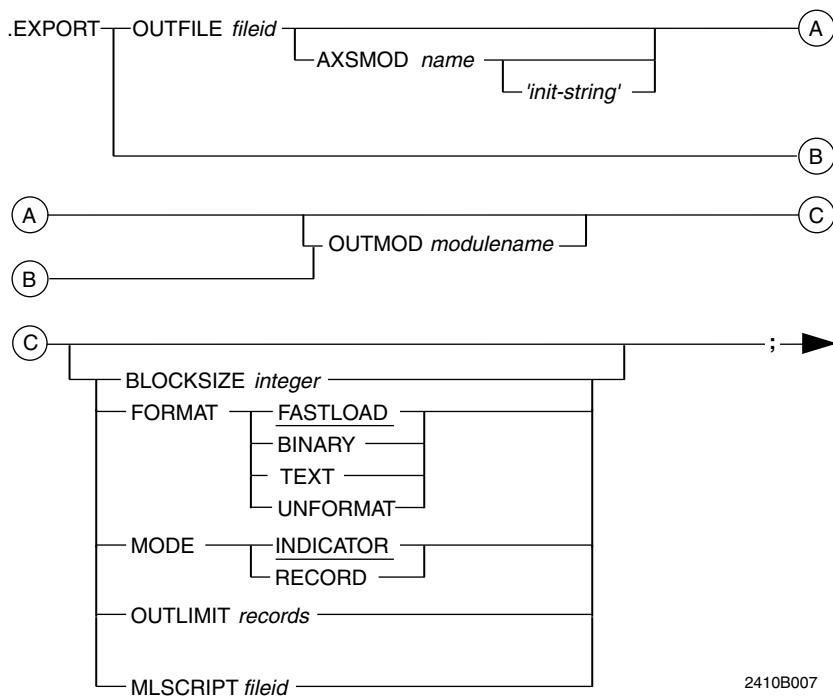
Topic	Usage Notes
Command Placement and Frequency	<p>The END EXPORT command must be the last command in a group of FastExport utility commands that specify an export task.</p> <p>Multiple END EXPORT commands can be used in a FastExport job script, but each export task specification that begins with a BEGIN EXPORT command must end with an END EXPORT command.</p>
Command Processing	<p>In response to the END EXPORT command, the FastExport utility sends a SELECT statement to the Teradata Database that:</p> <ul style="list-style-type: none"> <li>Places the resource locks on the Teradata Database tables</li> <li>Prepares the export data for return to the client system</li> </ul>

# EXPORT

## Purpose

The EXPORT command provides the client system destination and file format specifications for the export data retrieved from the Teradata Database and, optionally, generates a MultiLoad script file that can be used to reload the export data.

## Syntax



where

Syntax Element	Description
OUTFILE <i>fileid</i>	<p>Data destination file on the client system.</p> <p>The client system DD or equivalent statement specifies a file:</p> <ul style="list-style-type: none"> <li>In UNIX and Windows, the <i>fileid</i> is the path name for a file. If the path name has embedded white space characters, the entire path name must be enclosed in single or double quotes.</li> <li>In z/VM, the <i>fileid</i> is a FILEDEF name.</li> <li>In z/OS, the <i>fileid</i> is a DDNAME. (See the “z/OS <i>fileid</i> Usage Rules” topic in the “Usage Notes” subsection.)</li> </ul>

Syntax Element	Description
AXSMOD <i>name</i>	<p>Name of an access module file that exports data to a file</p> <p>To specify the OLE DB Access Module, use <i>oledb_axsmod.dll</i> on Windows platforms.</p> <p>The shared library file name may be selected if a custom access module exists.</p> <p><b>Note:</b> Large File Access Module is no longer available because the Data Connector API supports file sizes greater than 2 GB on Windows, HP-UX, IBM-AIX, and Solaris SPARC platforms.</p> <p>The AXSMOD option is not required for exporting to:</p> <ul style="list-style-type: none"> <li>• Disk files on either network-attached or channel-attached client systems</li> <li>• Magnetic tape files on channel-attached client systems</li> </ul> <p>It is required for exporting to magnetic tape and other types of files on network-attached systems.</p> <p>For more information about specific Teradata access modules, see <i>Teradata Tools and Utilities Access Module Reference</i>.</p>
' <i>init-string</i> '	[Optional] initialization string for the access module
OUTMOD <i>modulename</i>	<p>[Optional] user-written routine for processing the export data.</p> <p>In z/OS, <i>modulename</i> is the name of a load module. On UNIX and Windows platforms, it is the path name of the OUTMOD executable code file.</p> <p>FastExport provides six parameters to the named procedure, as described in <a href="#">“FastExport/OUTMOD Routine Interface”</a> on page 56.</p> <p><b>Note:</b> On some versions of UNIX, <i>./</i> prefix characters may have to be added to the OUTMOD <i>modulename</i> specification if the module is in the current directory.</p>
MODE...	<p>Format mode of the export data returned to the client system:</p> <ul style="list-style-type: none"> <li>• INDICATOR</li> <li>• RECORD</li> </ul> <p>The default, if a MODE option is not specified, is INDICATOR mode.</p> <p><b>Note:</b> FastExport does not support field mode. To export field mode data, use the appropriate format clauses in the SELECT statements to enable the Teradata Database to convert response data to character format.</p>

Syntax Element	Description
FORMAT...	<p>Record format of the export file on network-attached systems where:</p> <ul style="list-style-type: none"> <li>FASTLOAD specifies that each record is a two-byte integer, <i>n</i>, followed by <i>n</i> bytes of data, followed by an end-of-record marker, either X'0A' or X'0D'.</li> <li>BINARY specifies that each record is a two-byte integer, <i>n</i>, followed by <i>n</i> bytes of data.</li> <li>TEXT specifies that each record is an arbitrary number of bytes, followed by an end-of-record marker, which is a: <ul style="list-style-type: none"> <li>Line feed (X'0A') on UNIX platforms</li> <li>Carriage-return/line feed pair (X'0D0A') on Windows platforms</li> </ul> </li> <li>UNFORMAT specifies that each record is exported as it is received from CLIV2 without any client modifications.</li> </ul> <p><b>Note:</b> The FORMAT options apply only to UNIX and Windows platforms. The default, if a FORMAT option is not specified, is FASTLOAD.</p>
BLOCKSIZE <i>integer</i>	<p>Maximum block size that should be used when returning data to the client</p> <p>The default block size is 64K bytes, which is the maximum supported by the Teradata Database.</p> <p><b>Note:</b> The BLOCKSIZE specification for a FastExport EXPORT command cannot be larger than the row size supported by the Teradata Database.</p>
OUTLIMIT <i>records</i>	<p>Maximum number of response records that should be written to the output client file</p> <p>When this number is reached, the utility writes the following message to the print output file and stops processing response data:</p> <pre>Output limit of n exceeded.</pre>
MLSCRIPT <i>fileid</i>	<p>Destination file of the generated MultiLoad script file</p> <p>When the MLSCRIPT option is specified, FastExport generates a MultiLoad script file that can later be used to reload the export data back into the Teradata Database.</p> <p>The client system DD or equivalent statement specifies a file:</p> <ul style="list-style-type: none"> <li>In UNIX and Windows, the <i>fileid</i> is the pathname for a file</li> <li>In z/VM, the <i>fileid</i> is a FILEDEF name</li> <li>In z/OS, z/OS is a DDNAME. (See the “z/OS <i>fileid</i> Usage Rules” topic in the “Usage Notes” subsection.)</li> </ul> <p>By default, if the MLSCRIPT option is not specified, then the FastExport utility does not generate a MultiLoad script file.</p> <p><b>Note:</b> If the specified <i>fileid</i> already exists, it will be overwritten.</p>



## Usage Notes

Table 30 describes the things to consider when using the EXPORT command.

Table 30: EXPORT Command Usage Notes

Topic	Usage Notes
Access Module Release Level Compatibility	Release 07.03.00 of FastExport software is not compatible with access modules prepared for release 07.00.00 and earlier.
Attributes of the Destination File	<p>On channel-attached client systems, the attributes of the destination file must be compatible with the export data records that will be written there. (Compatibility is not a problem on network-attached UNIX and Windows client systems.)</p> <p>On channel-attached z/OS and z/VM systems, the attributes vary, depending on:</p> <ul style="list-style-type: none"> <li>Disposition of the file—If the execution of the FastExport utility is a restart operation, then the disposition of the destination file should be OLD</li> <li>Response mode—For all response modes, the attributes can specify any RECFM. However, RECFM=FB (fixed blocked) or RECFM=VB (variable blocked) are commonly used.</li> <li>Record length and block size—These must accommodate the specified format as shown in Table 31</li> </ul> <p>The Teradata Database data types in the channel-attached z/VM and z/OS environments are described in Table 32. Use this information to calculate the size of the exported data rows to assign appropriate values to the attributes of the destination file.</p>
Block Size Specification	<p>Two 64K-byte buffers are allocated for each session being used to transmit data from the Teradata Database to the client system.</p> <p>The minimum block size that must be allocated is one which will hold the largest possible parcel returned by the Teradata Database.</p> <p>If the specified block size is not large enough to hold the largest possible parcel, the Teradata Database returns an error to the SELECT statement and the utility is abnormally terminated.</p> <p>For a complete description of the parcel sizes, see:</p> <ul style="list-style-type: none"> <li><i>Teradata Call-Level Interface Version 2 Reference for Channel-Attached Systems</i></li> <li><i>Teradata Call-Level Interface Version 2 Reference for Network-Attached Systems</i></li> </ul>
Command Placement and Frequency	One EXPORT command is required for each export task in a FastExport job script. Place it anywhere between the BEGIN EXPORT command and the END EXPORT command that specify the export task.

Table 30: EXPORT Command Usage Notes (continued)

Topic	Usage Notes
MODE Specifications	<p>Both the INDICATOR and RECORD mode specifications return data in a client internal format with variable-length records:</p> <ul style="list-style-type: none"> <li>• Each record has a value for all of the columns specified by the SELECT statement</li> <li>• Variable-length columns are preceded by a two-byte control value indicating the length of the column data</li> <li>• Null columns have a value that is appropriate for the column data type</li> </ul> <p>Data records returned in indicator mode, however, have a set of bit flags that identify the columns that have a null value.</p> <p>For a complete description of these modes, see:</p> <ul style="list-style-type: none"> <li>• <i>Teradata Call-Level Interface Version 2 Reference for Channel-Attached Systems</i></li> <li>• <i>Teradata Call-Level Interface Version 2 Reference for Network-Attached Systems</i></li> </ul>
Multiple SELECT Statements	<p>If the export task specified multiple SELECT statements, the export data is returned in statement order. All response data for statement 1 is followed by the response data for statement 2, and so forth.</p> <p>If the same SELECT statement is executed multiple times, then the results of the first iteration are returned and processed before the second iteration of the SELECT statement is sent to the Teradata Database.</p>
SELECT Statement Processing	<p>If the export task specified multiple SELECT statements, the response data for all statements is returned in statement order—all response data for statement 1 will be first, followed by the data for statement 2, and so forth.</p> <p>If a single SELECT statement is executed multiple times, the results of the first iteration are returned and processed before the second SELECT statement is sent to the Teradata Database.</p>
z/OS <i>fileid</i> Usage Rules	<p>A DDNAME must obey the same construction rules as Teradata SQL column names except that:</p> <ul style="list-style-type: none"> <li>• The "at" character (@) is allowed as an alphabetic character</li> <li>• The underscore character (_) is not allowed</li> </ul> <p>The DDNAME must obey the applicable rules of the external system.</p> <p>If the DDNAME represents a data source on magnetic tape, the tape may be either labeled or nonlabeled, as supported by the operating system.</p>

[Table 31](#) describes the Record Length and Block Size Specification.

Table 31: Record Length and Block Size Specifications (Channel-Attached Client Systems)

RECFM	Description
FB	<p>LRECL must be exactly equal to the number of bytes of data being returned. The LRECL cannot be larger.</p> <p>For RECFM=FB, the BLKSIZE must also be a multiple of the LRECL. If not, records may be truncated, resulting in possible data integrity problems, or FastExport may append.</p> <p>Explicitly adding BLKSIZE to the JCL eliminates the possibility of using an invalid default BLKSIZE.</p>
VB	<p>Logical record length (LRECL) and block size (BLKSIZE) parameters should be large enough to accommodate the largest record that is anticipated.</p>
VBS or VS	<p>Maximum logical record length can exceed the physical length for a given data set.</p> <p>Spanned records, either blocked or unblocked, use a well-established straightforward protocol to break or segment records across blocks where necessary.</p> <p>While an individual segment never exceeds the length of a block, the logical record that it is a part of can span multiple blocks, and even volumes. Thus spanned records are the only way to create output files with rows whose length exceed the 32K-byte block size, up to the maximum of 64K bytes that is supported by the Teradata Database.</p> <p>Even though the maximum LRECL that can be specified with JCL is 32,760, there is no practical limit on the actual length of spanned records.</p> <p>For output consisting of records exceeding this maximum LRECL (greater than approximately 32K, for example), simply specify LRECL=X. There is no other special JCL requirement for creating such records when using the VBS or VS record format.</p> <p>Always specify the BLKSIZE according to the performance characteristics of the target device or media. This usually means specifying the largest possible BLKSIZE.</p> <p>In some cases, the performance of the FastExport utility may be improved by specifying RECFM=VBS when:</p> <ul style="list-style-type: none"> <li>• The largest row is appreciably smaller than 32K bytes in length</li> <li>• There is a large variation in row sizes</li> </ul> <p>The spanned/blocked format maximizes data packing. Because fewer blocks are required to convey the same number of logical records, the FastExport job runs quicker.</p>

Table 31: Record Length and Block Size Specifications (Channel-Attached Client Systems) (continued)

RECFM	Description
	<p>For example, assuming a block size of 32,756 bytes:</p> <ul style="list-style-type: none"> <li>Using RECFM=VB, a 20,000-byte record and a 4,000-byte record could be packed into a newly created block. But, if the next record were 12,000 bytes long it clearly would exceed the length of the block and would have to be packed into the following block.</li> <li>Using RECFM=VBS, the 12,000-byte record could be segmented such that the first 8,740 bytes could be packed into the original block and the remaining 3,260 bytes packed into the subsequent block—taking into account that there must be one 4-byte Block Descriptor Word (BDW) per block and one 4-byte Segment Descriptor Word (SDW) per segment; and a segment must be fully contained within a block</li> </ul>

**Note:** A FastExport job will fail with an Error 1776 if rows greater than 32K bytes are exported using a RECFM= specification other than VBS or VS.

Note also that not all applications can read spanned data records. Always make sure that applications support spanned records before specifying these formats.

**Example:** This DD statement requests spanned records for a FastExport EXPORT to fileid named OUTPUT:

```
//OUTPUT DD DSN=ASG.FEXP.Z, DISP=(NEW,CATLG),
//      DCB=(RECFM=VBS,LRECL=32760, BLKSIZE=32756, DSORG=PS),
//      UNIT=SYSDA, SPACE=(CYL,(100,20))
```

Table 32 contains the Data Type description for default Channel-Attached Client Systems.

Table 32: Data Type Descriptions (Channel-Attached Client Systems)

Data Type	Output Length	Description
BYTE( <i>n</i> )	<i>n</i> bytes	<i>n</i> bytes
BYTEINT	1 byte	8-bit signed binary
CHAR( <i>n</i> ) CHARS( <i>n</i> ) CHARACTERS( <i>n</i> )	<i>n</i> bytes	<i>n</i> EBCDIC characters
DATE	4 bytes	<p>32-bit integer in the internal date format of the Teradata Database.</p> <p>For details, see the Teradata <i>Database Design</i> and <i>SQL Data Types and Literals</i> reference documentation.</p> <p><b>Note:</b> If a DATEFORM command has been used to specify ANSIDATE as the DATE data type, the FastExport utility internally converts each DATE data type to a CHAR(10) field.</p>

Table 32: Data Type Descriptions (Channel-Attached Client Systems) (continued)

Data Type	Output Length	Description
DECIMAL $x$ DECIMAL( $x$ ) DECIMAL( $x,y$ )	( $x+1$ ) / 2 bytes	$x$ packed decimal digits and sign
FLOAT FLOATING	8 bytes	64-bit (double-precision) floating point
GEOSPATIAL DATA	maximum 64000	FastExport does not support Geospatial data represented by LOBs.
INTEGER	4 bytes	32-bit signed binary
LONG VARCHAR	$m+2$ characters where $m \leq n$	Same as VARCHAR (32000) characters
Fixed Length Period Data Types: PERIOD(DATE) PERIOD(TIME( $n$ )) PERIOD(TIME( $n$ ) WITH TIME ZONE)	max=8 byte max=12 bytes max=16 bytes	The precision specified must be $0 < n < 6$ : precision=0 (n/a) precision= $n$ precision= $n$ For details, see the Teradata <i>Database Design and SQL Data Types and Literals</i> reference documentation.
Variable Length Period Data Types: PERIOD(TIMESTAMP( $n$ )) PERIOD(TIMESTAMP( $n$ ) WITH TIME ZONE)	max=20 bytes max=24 bytes	The precision specified must be $0 < n < 6$ : precision= $n$ precision= $n$ For details, see the Teradata <i>Database Design and SQL Data Types and Literals</i> reference documentation.
SMALLINT	2 bytes	16-bit signed binary
VARBYTE( $n$ )	$m+2$ bytes where $m \leq n$	16-bit integer, count $m$ , followed by $m$ bytes of data
VARCHAR( $n$ )	$m+2$ bytes where $m \leq n$	16-bit integer, count $m$ , followed by $m$ EBCDIC characters

Refer to *SQL Data Types and Literals* for more information.

## Example 1 Using the OUTFILE and FORMAT Specifications

The following example specifies that the exported records to be loaded are written to `/home/fexpuser/tests/out1` and that the format of each record is unformat:

```
.EXPORT OUTFILE /home/fexpuser/tests/out1
FORMAT UNFORMAT ;
```

## Example 2 Specifying an OUTMOD Routine

The following example for a UNIX client system runs an OUTMOD routine that has been compiled and linked as *feomod.so*:

```
.EXPORT OUTMOD ./feomod.so;
```

The following example for a Windows client system runs the same OUTMOD routine that has been compiled and linked as *feomod.dll*:

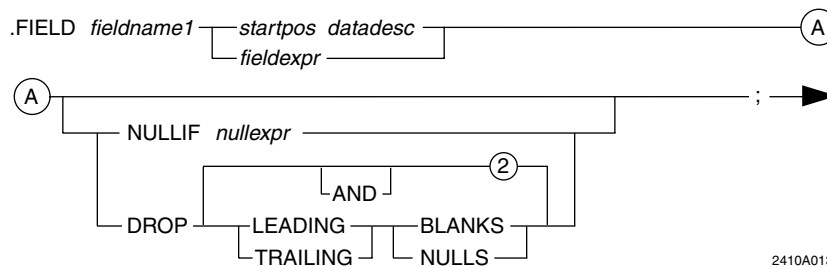
```
.EXPORT OUTMOD ./feomod.dll;
```

# FIELD

## Purpose

The FIELD command specifies a field of the input record that provides data values for the constraint parameters of the SELECT statement. Each field defined by a FIELD command is sent to the Teradata Database as part of the data record containing data values defined by a USING modifier for the SELECT statement.

## Syntax



where

Syntax Element	Description
<i>fieldname1</i>	Name of an input record field that is referenced by a variable parameter name in the WHERE condition of the SELECT statement
<i>startpos</i>	Starting position of the field in an input data record <i>startpos</i> can be specified as an: <ul style="list-style-type: none"> <li>• Unsigned integer, which is a character position starting with 1</li> <li>• Asterisk (*), which means the next available character position beyond the preceding field</li> </ul> <b>Note:</b> When using the CONTINUEIF condition of the LAYOUT command to continue input records, a <i>startpos</i> specified by an integer value refers to a character position in the final concatenated record from which the continuation indicator has been removed.
<i>datadesc</i>	Type and length of data in the field This description is used to generate the data description for this field in the USING modifier for the SELECT statement. The <i>datadesc</i> specification can be any of the data type phrases shown in the <i>SQL Data Types and Literals</i> reference documentation.

Syntax Element	Description
<i>fieldexpr</i>	<p>Concatenation of two or more items, either fields or character constants or string constants or a combination of these in the following form:</p> <p><i>fieldname2</i>    <i>fieldname2</i>    <i>fieldname2</i> ...</p> <p>Nested concatenations are not supported. Each <i>fieldname2</i> that is actually a field by its own FIELD command must be defined.</p> <p>Valid character and string constants are as described in the Teradata <i>SQL Fundamentals</i> documentation.</p>
NULLIF <i>nullexpr</i>	<p>Condition used for selectively inserting a null value into the affected column</p> <p>The condition is specified as a conditional expression involving any number of fields, each represented by its <i>fieldname</i> and constants.</p> <p>Each <i>fieldname</i> appearing in the conditional expression must be defined by either:</p> <ul style="list-style-type: none"> <li>• The <i>startpos</i> and <i>datadesc</i> parameters of the FIELD command</li> <li>• A FILLER command</li> </ul>
DROP...	<p>Character positions to be dropped from the <i>fieldname1</i></p> <p>These must be of a character data type.</p>

## Usage Notes

Table 33 describes the things to consider when using the FIELD command.

Table 33: FIELD Command Usage Notes

Topic	Usage Notes
Command Placement and Frequency	<p>A FIELD command must be preceded by a LAYOUT command.</p> <p>One or more FIELD commands, or a combination of FIELD command and FILLER command, define the composition of the input data record to supply values for the USING modifier of the SELECT statement.</p>
Specifying DECIMAL Data Types	<p>The following input length and field descriptions apply for the DECIMAL data type specifications which make in the <i>datadesc</i> parameter.</p> <p>DECIMAL (<i>x</i>) and DECIMAL (<i>x,y</i>)</p> <ul style="list-style-type: none"> <li>• Length: 1, 2, 4, 8, or 16 bytes (network); packed decimal (mainframe)</li> <li>• Description: 128-bit double precision floating point</li> </ul> <p>For more information on the DECIMAL data type, see <i>SQL Data Types and Literals</i>.</p>



Table 33: FIELD Command Usage Notes (continued)

Topic	Usage Notes
Specifying Period Data Types	<p>A period is an anchored duration. It represents a set of contiguous time granules within that duration. A period is implemented using a Period data type. Each period consists of two elements:</p> <ul style="list-style-type: none"> <li>• BEGIN (the beginning element)</li> <li>• END (the ending element)</li> </ul> <p>The element type is one of the following DateTime data types.</p> <ul style="list-style-type: none"> <li>• PERIOD(DATE)</li> <li>• PERIOD(TIME[(n)])</li> <li>• PERIOD(TIME[(n)] WITH TIME ZONE)</li> <li>• PERIOD(TIMESTAMP[(n)])</li> <li>• PERIOD(TIMESTAMP[(n)] WITH TIME ZONE)</li> </ul> <p>For more information on the PERIOD data type, see the <i>SQL Data Types and Literals</i> book.</p>
Using ANSI/SQL DateTime Data Types	<p>When the DATEFORM command is used to specify ANSIDATE as the DATE data type, FastExport internally converts each DATE field to a CHAR(10) field. All ANSI/SQL DateTime TIME, TIMESTAMP, and INTERVAL data types must be converted to fixed-length CHAR data types to specify column/field names in a FastExport FIELD command.</p> <p><a href="#">Table 34</a> provides the conversion specifications and format examples for each ANSI/SQL DateTime specification.</p>

[Table 34](#) describes the ANSI/SQL Date Time Specifications.

Table 34: ANSI/SQL DateTime Specifications

<b>DATE</b>	
Convert to:	CHAR(10)
Format:	yyyy/mm/dd
Example:	1998/01/01
<b>TIME</b>	
<b>TIME (n)</b>	
Where <i>n</i> is the number of digits after the decimal point, 0 through 6. (Default = 6.)	
Convert to:	CHAR(8 + <i>n</i> + (1 if <i>n</i> > 0, otherwise 0))
Format ( <i>n</i> = 0):	hh:mm:ss
Example:	11:37:58
Format ( <i>n</i> = 4):	hh:mm:ss.ssss
Example:	11:37:58.1234
<b>TIMESTAMP</b>	
<b>TIMESTAMP (n)</b>	

Table 34: ANSI/SQL DateTime Specifications (continued)

Where  $n$  is the number of digits after the decimal point, 0 through 6. (Default = 6.)

Convert to: CHAR(19 +  $n$  + (1 if  $n > 0$ , otherwise 0))

Format ( $n = 0$ ): *yyyy-mm-dd hh:mm:ss*

Example: 1998-09-04 11:37:58

Format ( $n = 4$ ): *yyyy-mm-dd hh:mm:ss.ssss*

Example: 1998-09-04 11:37:58.1234

**TIME WITH TIME ZONE  
TIME ( $n$ ) WITH TIME ZONE**

Where  $n$  is the number of digits after the decimal point, 0 through 6. (Default = 6.)

Convert to: CHAR(14 +  $n$  + (1 if  $n > 0$ , otherwise 0))

Format ( $n = 0$ ): *hh:mm:ss{±}hh:mm*

Example: 11:37:58-08:00

Format ( $n = 4$ ): *hh:mm:ss.ssss {±} hh:mm*

Example: 11:37:58.1234-08:00

**TIMESTAMP WITH TIME ZONE  
TIMESTAMP ( $n$ ) WITH TIME ZONE**

Where  $n$  is the number of digits after the decimal point, 0 through 6. (Default = 6.)

Convert to: CHAR(25 +  $n$  + (1 if  $n > 0$ , otherwise 0))

Format ( $n = 0$ ): *yyyy-mm-dd hh:mm:ss{±}hh:mm*

Example: 1998-09-24 11:37:58+07:00

Format ( $n = 4$ ): *yyyy-mm-dd hh:mm:ss.ssss{±}hh:mm*

Example: 1998-09-24 11:37:58.1234+07:00

**INTERVAL YEAR  
INTERVAL YEAR ( $n$ )**

Where  $n$  is the number of digits, 1 through 4. (Default = 2.)

Convert to: CHAR( $n$ )

Format ( $n = 2$ ): *yy*

Example: 98

Format ( $n = 4$ ): *yyyy*

Example: 1998

**INTERVAL YEAR TO MONTH  
INTERVAL YEAR ( $n$ ) TO MONTH**

Where  $n$  is the number of digits, 1 through 4. (Default = 2.)

Convert to: CHAR( $n + 3$ )

Format ( $n = 2$ ): *yy-mm*

Example: 98-12

Table 34: ANSI/SQL DateTime Specifications (continued)

Format ( $n = 4$ ):	<i>yyyy-mm</i>
Example:	1998-12

---

**INTERVAL MONTH**  
**INTERVAL MONTH ( $n$ )**

Where  $n$  is the number of digits, 1 through 4. (Default = 2.)

Convert to:	CHAR( $n$ )
-------------	-------------

Format ( $n = 2$ ):	<i>mm</i>
Example:	12

Format ( $n = 4$ ):	<i>mmmm</i>
Example:	0012

---

**INTERVAL DAY**  
**INTERVAL DAY ( $n$ )**

Where  $n$  is the number of digits, 1 through 4. (Default = 2.)

Convert to:	CHAR( $n$ )
-------------	-------------

Format ( $n = 2$ ):	<i>dd</i>
Example:	31

Format ( $n = 4$ ):	<i>dddd</i>
Example:	0031

---

**INTERVAL DAY TO HOUR**  
**INTERVAL DAY ( $n$ ) TO HOUR**

Where  $n$  is the number of digits, 1 through 4. (Default = 2.)

Convert to:	CHAR( $n + 3$ )
-------------	-----------------

Format ( $n = 2$ ):	<i>dd hh</i>
Example:	31 12

Format ( $n = 4$ ):	<i>dddd hh</i>
Example:	0031 12

---

**INTERVAL DAY TO MINUTE**  
**INTERVAL DAY ( $n$ ) TO MINUTE**

Where  $n$  is the number of digits, 1 through 4. (Default = 2.)

Convert to:	CHAR( $n + 6$ )
-------------	-----------------

Format ( $n = 2$ ):	<i>dd hh:mm</i>
Example:	31 12:59

Format ( $n = 4$ ):	<i>dddd hh:mm</i>
Example:	0031 12:59

---

Table 34: ANSI/SQL DateTime Specifications (continued)

**INTERVAL DAY TO SECOND**  
**INTERVAL DAY (*n*) TO SECOND**  
**INTERVAL DAY TO SECOND (*m*)**  
**INTERVAL DAY (*n*) TO SECOND (*m*)**

Where

- *n* is the number of digits, 1 through 4. (Default = 2.)
- *m* is the number of digits after the decimal point, 0 through 6. (Default = 6.)

Convert to: CHAR( $n + 9 + m + (1 \text{ if } m > 0, 0 \text{ otherwise})$ )

Format ( $n = 2, m = 0$ ): *dd hh:mm:ss*  
 Example: 31 12:59:59

Format ( $n = 4, m = 4$ ): *dddd hh:mm:ss.ssss*  
 Example: 0031 12:59:59:59.1234

---

**INTERVAL HOUR**  
**INTERVAL HOUR (*n*)**

Where *n* is the number of digits, 1 through 4. (Default = 2.)

Convert to: CHAR(*n*)

Format ( $n = 2$ ): *hh*  
 Example: 12

Format ( $n = 4$ ): *hhhh*  
 Example: 0012

---

**INTERVAL HOUR TO MINUTE**  
**INTERVAL HOUR (*n*) TO MINUTE**

Where *n* is the number of digits, 1 through 4. (Default = 2.)

Convert to: CHAR( $n + 3$ )

Format ( $n = 2$ ): *hh:mm*  
 Example: 12:59

Format ( $n = 4$ ): *hhhh:mm*  
 Example: 0012:59

---

**INTERVAL HOUR TO SECOND**  
**INTERVAL HOUR (*n*) TO SECOND**  
**INTERVAL HOUR TO SECOND (*m*)**  
**INTERVAL HOUR (*n*) TO SECOND (*m*)**

Where

- *n* is the number of digits, 1 through 4. (Default = 2.)
- *m* is the number of digits after the decimal point, 0 through 6. (Default = 6.)

Convert to: CHAR( $n + 6 + m + (1 \text{ if } m > 0, 0 \text{ otherwise})$ )

Format ( $n = 2, m = 0$ ): *hh:mm:ss*  
 Example: 12:59:59

Table 34: ANSI/SQL DateTime Specifications (continued)

Format ( $n = 4, m = 4$ ): *hhhh:mm:ss.ssss*  
 Example: 0012:59:59.1234

---

**INTERVAL MINUTE**  
**INTERVAL MINUTE ( $n$ )**

Where  $n$  is the number of digits, 1 through 4. (Default = 2.)

Convert to: CHAR( $n$ )

Format ( $n = 2$ ): *mm*

Example: 59

Format ( $n = 4$ ): *mmmm*

Example: 0059

---

**INTERVAL MINUTE TO SECOND**  
**INTERVAL MINUTE ( $n$ ) TO SECOND**  
**INTERVAL MINUTE TO SECOND ( $m$ )**  
**INTERVAL MINUTE ( $n$ ) TO SECOND ( $m$ )**

Where

- $n$  is the number of digits, 1 through 4. (Default = 2.)
- $m$  is the number of digits after the decimal point, 0 through 6. (Default = 6.)

Convert to: CHAR( $n + 3 + m + (1 \text{ if } m > 0, 0 \text{ otherwise})$ )

Format ( $n = 2, m = 0$ ): *mm:ss*

Example: 59:59

---

Format ( $n = 4, m = 4$ ): *mmmm:ss.ssss*

Example: 0059:59.1234

---

**INTERVAL SECOND**  
**INTERVAL SECOND ( $n$ )**  
**INTERVAL SECOND ( $n,m$ )**

Where

- $n$  is the number of digits, 1 through 4. (Default = 2.)
- $m$  is the number of digits after the decimal point, 0 through 6. (Default = 6.)

Convert to: CHAR( $n + m + (1 \text{ if } m > 0, 0 \text{ otherwise})$ )

Format ( $n = 2, m = 0$ ): *ss*

Example: 59

Format ( $n = 4, m = 4$ ): *ssss.ssss*

Example: 0059.1234

---

# FILLER

## Purpose

The FILLER command specifies a field that is *not* sent to the Teradata Database as part of the input record that provides data values for the constraint parameters of the SELECT statement.

## Syntax

```
.FILLER fieldname startpos datadesc ;
```

2409A030

where

Syntax Element	Description
<i>fieldname</i>	Optional name for the input record field The <i>fieldname</i> specification is required only if the field is referred to by the <i>nullexpr</i> condition of a FIELD command.
<i>startpos</i>	Starting position of the specified field in an input data record <i>startpos</i> can be specified as an: <ul style="list-style-type: none"> <li>• Unsigned integer, which is the character position starting with 1</li> <li>• Asterisk (*), which means the next available character position beyond the preceding field</li> </ul> <b>Note:</b> When using the CONTINUEIF condition of the LAYOUT command to continue input records, a <i>startpos</i> specified by an integer value refers to a character position in the final concatenated record from which the continuation indicator has been removed.
<i>datadesc</i>	Type and length of data in the field The <i>datadesc</i> specification can be any of the data type phrases shown in the Teradata <i>Utilities</i> reference documentation. This description is used to generate the data description for this field in the USING modifier for the SELECT statement.

## Usage Notes

Table 35 describes the things to consider when using the FILLER command.

Table 35: FILLER Command Usage Notes

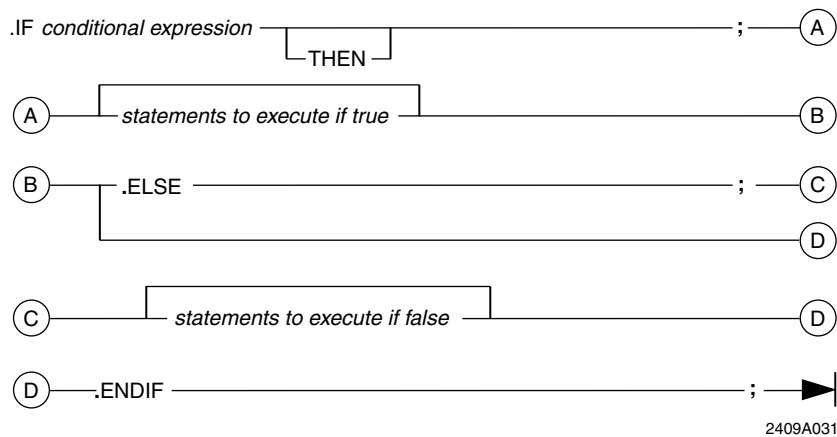
Topic	Usage Notes
Command Placement and Frequency	<p>A FILLER command must be preceded by a LAYOUT command.</p> <p>One or more FILLER commands, or a combination of FILLER commands and FIELD commands, define the composition of the input data record to supply values for the USING modifier of the SELECT statement.</p>

# IF, ELSE, and ENDIF

## Purpose

The IF, ELSE, and ENDIF commands provide conditional control of execution processes.

## Syntax



## Usage Notes

[Table 36](#) describes the things to consider when using the IF, ELSE, and ENDIF commands.

Table 36: IF, ELSE and END IF Command Usage Notes

Topic	Usage Notes
ELSE Clause	Use the optional ELSE clause to execute commands when the condition is evaluated as false.
Nesting IF Commands	FastExport supports the nesting of IF commands to a level of 100.
Numeric Results of the Conditional Expression	The <i>conditional expression</i> is an expression that can be evaluated as either true or false. When evaluation of the expression returns a numeric result: <ul style="list-style-type: none"> <li>• Zero is interpreted as false</li> <li>• Nonzero results are interpreted as true</li> </ul>
Variables in the IF Expression	The <i>conditional expression</i> can be either user-defined variables or predefined system variables.



Table 36: IF, ELSE and END IF Command Usage Notes (continued)

Topic	Usage Notes
Variable Substitutions	<p>Any ELSE or ENDIF commands must be present in their entirety and cannot be composed simply of variables in need of substitution.</p> <p>Commands and statements following an IF, ELSE, or ENDIF structure that are not executed are not parsed and do not have their variables substituted.</p>

## Example 1

FastExport is case sensitive when comparing &SYS system variables. In this example, the RUN FILE command does not execute because the substituted values returned are all capitals:

```
0003 .IF '&SYSDAY' = 'Fri' THEN;
14:10:28 - FRI MAY 09, 1993
UTY2402 Previous statement modified to:
0004 .IF 'FRI' = 'Fri' THEN;
0005.RUN FILE UTNTS38;
0006 .ENDIF;
```

Always consider this factor when creating a script to force the execution of a predetermined sequence of events. If 'FRI' is substituted in line 0003, the compare would work and the RUN FILE command would execute.

## Example 2

In the following example, the user has created the table named &TABLE and a variable named CREATERC, into which is set the system return code resulting from the execution of the CREATE TABLE statement: .SET CREATERC TO &SYSRC;

```
.SET CREATERC TO &SYSRC;
.IF &CREATERC = 3803 /* Table &TABLE exists */ THEN;
.RUN FILE RUN01;
.ELSE;
.IF &CREATERC <> 0 THEN;
.LOGOFF &CREATRC;
.ENDIF;
.ENDIF:
```

If the table name has not already been used, and the return code is not zero, the return code evaluates to an error condition and the job logs off with the error code displayed.

# IMPORT

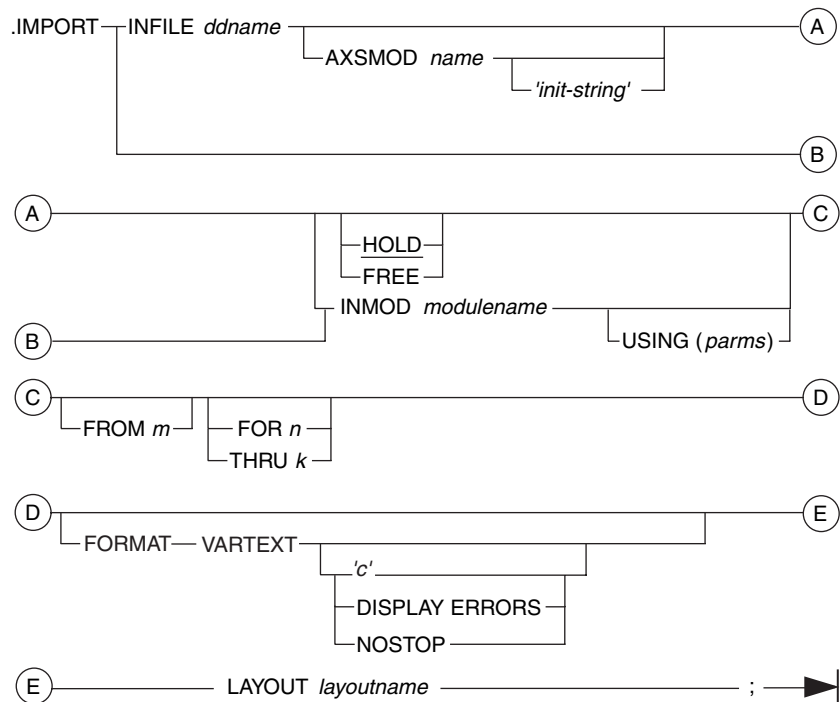
## Purpose

The IMPORT command defines the client file that provides the USING data values for the FastExport SELECT statement.

## Syntax

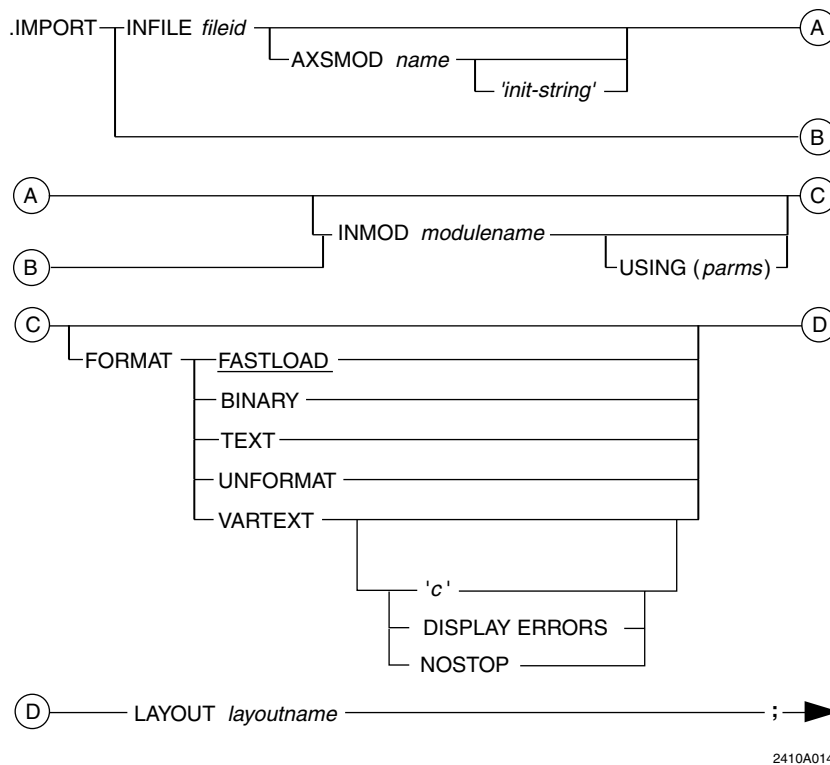
The IMPORT command syntax depends on whether the FastExport utility is running on a channel-attached or network-attached client system. Several of the syntax elements are common to both configurations, while others are specific to each.

### For Channel-Attached Client Systems



2410C006

**For Network-Attached Client Systems**



where

Syntax Element	Description
INFILE <i>fileid</i>	Input file on the client system The client system DD or equivalent statement specifies a file: <ul style="list-style-type: none"> <li>In UNIX and Windows, the <i>fileid</i> is the path name for a file If the path name has embedded white space characters, the path name must be enclosed in single or double quotes.</li> <li>In z/VM, the <i>fileid</i> is a FILEDEF name</li> <li>In z/OS, the <i>fileid</i> is a DDNAME. (See the “z/OS <i>fileid</i> Usage Rules” topic in the “Usage Notes” subsection.)</li> </ul>

Syntax Element	Description
AXSMOD <i>name</i>	<p>Name of the access module file to be used to import data, where the Named Pipes Access Module is:</p> <ul style="list-style-type: none"> <li><i>np_axsmod.sl</i> on HP-UX platforms</li> <li><i>np_axsmod.so</i> on MP-RAS; IBM-AIX; Solaris SPARC and Solaris Opteron platforms</li> <li><i>np_axsmod.dll</i> on Windows platforms</li> </ul> <p>A shared library file name can be used if a custom access module exists.</p> <p><b>Note:</b> Large File Access Module is no longer available because the Data Connector API supports file sizes greater than 2 gigabytes on Windows, HP-UX, IBM-AIX, and Solaris SPARC platforms.</p> <p>The AXSMOD option is not required for importing from:</p> <ul style="list-style-type: none"> <li>Disk files on either network-attached or channel-attached client systems</li> <li>Magnetic tape files on channel-attached client systems</li> </ul> <p>It is required for importing from magnetic tape and other types of files on network-attached client systems.</p> <p>For more information about specific Teradata access modules, see <i>Teradata Tools and Utilities Access Module Reference</i>.</p>
<i>init-string</i>	Optional initialization string for the access module
INMOD <i>modulename</i>	<p>Optional user-written routine for preprocessing the input data</p> <p>In z/OS, <i>modulename</i> is the name of a load module. On UNIX and Windows client systems, it is the pathname for the INMOD executable code file.</p> <p>When both the INFILE <i>fileid</i> and the INMOD <i>modulename</i> parameters are specified, FastExport reads the input file and passes the data to the INMOD routine for preprocessing.</p> <p>If the INFILE <i>fileid</i> parameter is not specified, FastExport expects the INMOD routine to provide the input data record.</p> <p>FastExport provides two parameters to the named routine, as described in <a href="#">“FastExport/INMOD Routine Interface” on page 54</a>.</p> <p><b>Note:</b> On some versions of UNIX, <i>./</i> prefix characters may have to be added to the INMOD <i>modulename</i> specification if the module is in the current directory.</p>

Syntax Element	Description
USING ( <i>parms</i> )	<p>Character string containing parameters can be passed to the INMOD routine:</p> <ul style="list-style-type: none"> <li>The <i>parms</i> string can include one or more character strings, each delimited on either end by either an apostrophe or a quotation mark</li> <li>The maximum size of the <i>parms</i> string is 1K bytes</li> <li>Parentheses within delimited character strings have the same syntactical significance as alphabetic characters</li> <li>Before passing the <i>parms</i> string to the INMOD routine, FastExport replaces the following with a single blank character: <ul style="list-style-type: none"> <li>Each comment</li> <li>Each consecutive sequence of white space characters, such as blank, tab, and so on, that appears outside of delimited strings</li> </ul> </li> <li>The entire <i>parms</i> string must be enclosed in parentheses and, on channel-attached client systems, the parentheses are included in the string passed to the INMOD routine</li> </ul> <p><b>Note:</b> The <i>parms</i> string must be FDLINMOD for INMOD routines written for the prior Pascal version of FastLoad (program FASTMAIN).</p>
FORMAT...	<p>Record format of the input file, where:</p> <ul style="list-style-type: none"> <li>FASTLOAD specifies that each record is a two-byte integer, <i>n</i>, followed by <i>n</i> bytes of data, followed by an end-of-record marker, either X '0A' or X '0D'</li> <li>BINARY specifies that each record is a two-byte integer, <i>n</i>, followed by <i>n</i> bytes of data</li> <li>TEXT specifies that each record is an arbitrary number of bytes followed by an end-of-record marker, either X '0A' or X '0D'</li> </ul> <p><b>Note:</b> TEXT format does not support numeric data. Do not specify TEXT if the MLSCRIPT option of an EXPORT command is also used.</p> <ul style="list-style-type: none"> <li>UNFORMAT specifies that each record is imported as it is received from CLIV2 without any client modifications</li> <li>VARTEXT specifies that each record is in variable-length text record format, with each field separated by a delimiter character</li> </ul> <p><b>Note:</b> All above FORMAT options apply to UNIX and Windows platforms. The VARTEXT option applies to mainframe. The default FORMAT option for UNIX and Windows platforms is FASTLOAD. The default FORMAT for mainframe is “use record boundaries “, meaning the input data is read record-by-record and the LAYOUT is applied to each record.</p>
'c'	<p>[Optional] Specification of the delimiter character that separates fields in the variable-length text records of the input data source</p> <p>The default, if a 'c' specification is not used, is the pipe character (   ).</p>
DISPLAY ERRORS	[Optional] Keyword specification that writes input data records that produce errors to the standard error file
NOSTOP	[Optional] Keyword specification that inhibits the FastExport termination in response to an error condition associated with a variable-length text record
LAYOUT layoutname	Identifier of the file layout description, as specified by a prior LAYOUT command

## Usage Notes

[Table 37](#) describes the things to consider when using the IMPORT command.

Table 37: IMPORT Command Usage Notes

Topic	Usage Notes
Access Module Release Level Compatibility	Release 07.03.00 and later of the FastExport utility software is not compatible with access modules prepared for release 07.00.00 and earlier.
Command Frequency and Placement	If the export task uses a LAYOUT command, then an IMPORT command is required, and it must appear after the LAYOUT command.
Data Type Specifications	When using the VARTEXT specification, VARCHAR, VARBYTE, and LONG VARCHAR are the only valid data type specifications which can be used in the FastExport layout FIELD command and FILLER command.  For additional information on data types, see <a href="#">Table 32 on page 84</a> .
Error Record Handling	When FastExport encounters an error condition in an input record, it normally discards the record and terminates. In loading variable-length text records, either or both of these functions can be inhibited by specifying the options: <ul style="list-style-type: none"> <li>• DISPLAY ERRORS</li> <li>• NOSTOP</li> </ul> By specifying both options and redirecting STDERR to a file location instead of the terminal screen, the FastExport job will run to completion and save all the error records. Then it can be manually modify and loaded.
Input Record Requirements	The total number of fields in each input record must be equal to or greater than the number of fields described in the FastExport layout FIELD command and FILLER command.  If it is less, FastExport generates an error message. If it is more, the Teradata Database ignores the extra fields.  The last field of a record does not have to end with a delimiter character. It can end with a delimiter character, but it is not required.
Multiple Physical Records	If the FastExport task reads the input file and constructs a logical record from multiple physical records, this is performed before the physical record is passed to the INMOD routine. The INMOD routine is invoked only one time for the generation of each USING record.
Null Fields	Two consecutive delimiter characters direct FastExport to null the field corresponding to the one right after the first delimiter character.  Also, if the last character in a record is a delimiter character, and there is at least one more field to be processed, then FastExport nulls the field corresponding to the next one to be processed, as defined in the layout FIELD command and FILLER command.

Table 37: IMPORT Command Usage Notes (continued)

Topic	Usage Notes
VARTEXT Records	When VARTEXT is specified, FastExport assumes that the input data is variable-length text fields separated by a field-delimiter character. The utility parses each input data record on a field-by-field basis, and creates a VARCHAR field for each input text field.
z/OS <i>fileid</i> Usage Rules	<p>A DDNAME must obey the same construction rules as Teradata SQL column names except that:</p> <ul style="list-style-type: none"> <li>• The “at” character (@) is allowed as an alphabetic character</li> <li>• The underscore character (_) is not allowed</li> </ul> <p>The DDNAME must obey the applicable rules of the external system.</p> <p>If the DDNAME represents a data source on magnetic tape, the tape may be either labeled or nonlabeled, as supported by the operating system.</p>

### Example 1 Using the INFILE and FORMAT Specifications

This example specifies that the using data for the SELECT statement is contained in */home/fexpuser/tests/data1* and that the format of each record is binary.

```
.IMPORT INFILE /home/fexpuser/tests/data1
FORMAT BINARY
Layout lay1;
```

### Example 2 Specifying an INMOD Routine

The following example for a UNIX client system runs an INMOD routine that has been compiled and linked as *feimod.so*:

```
.IMPORT INMOD ./feimod.so LAYOUT lay1;
```

The following example for a Windows client system runs the same INMOD routine that has been compiled and linked as *feimod.dll*:

```
.IMPORT INMOD ./feimod.dll LAYOUT lay1;
```

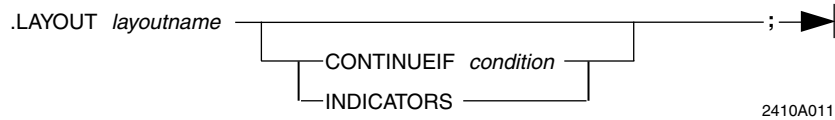
---

# LAYOUT

## Purpose

The LAYOUT command, used with an immediately following sequence of FIELD and FILLER commands, specifies the layout of the file that provides data values for the USING modifier of the SELECT statement.

## Syntax



where

Syntax Element	Description
<i>layoutname</i>	Name assigned to the layout for reference by one or more subsequent IMPORT commands  A <i>layoutname</i> must obey the same construction rules as Teradata SQL column names.



Syntax Element	Description
CONTINUEIF <i>condition</i>	<p>Conditional phrase in which <i>condition</i> is of the form:</p> <p style="padding-left: 40px;"><i>position = value</i></p> <p>where:</p> <ul style="list-style-type: none"> <li><i>position</i> is an unsigned integer (never an asterisk) that specifies the starting character position of the field of every input record that contains the continuation indicator</li> <li><i>value</i> is the continuation indicator specified as a character constant or a string constant. FastExport uses the length of the constant as the length of the continuation indicator field.</li> </ul> <p><b>Note:</b> The condition specified as <i>position = value</i> is case sensitive—always specify the correct character case for this parameter.</p> <p>If the conditional phrase is true, then FastExport forms a single record by concatenating the next input record at the end of the current record. (The current record is the one most recently obtained from the external data source.)</p> <p>If the conditional phrase is false, then FastExport uses the current input record either by itself or as the last of a sequence of concatenated records.</p> <p><b>Note:</b> Regardless of whether the condition evaluates to true or false, FastExport removes the tested string (the continuation indicator field) from each record.</p>
INDICATORS	<p>Condition that the input records defined by this LAYOUT command are in indicator mode</p> <p>That is, the first <i>n</i> bytes of each record are indicator bytes, where <i>n</i> is the rounded up integer quotient of the number of fields defined by the LAYOUT command, divided by 8.</p> <p>If this option is specified, the following FIELD commands must accurately define each field of the input record. The number of the defined fields and fillers is used to calculate the number of bytes of indicator data that are in each input record.</p>

## Usage Notes

Table 38 describes the things to consider when using the LAYOUT command.

Table 38: LAYOUT Command Usage Notes

Topic	Usage Notes
Command Frequency and Placement	A LAYOUT command specification must be referenced by each IMPORT command in the FastExport job script. In all cases, the LAYOUT command must be presented before an IMPORT command that references it. Each LAYOUT command must be immediately followed by a series of FIELD and FILLER commands that define the composition of a logical record.

Table 38: LAYOUT Command Usage Notes (continued)

Topic	Usage Notes
Using the same LAYOUT command in Multiple FastExport Tasks	The same <i>layoutname</i> specification can be referenced in more than one FastExport task, provided that: <ul style="list-style-type: none"><li>• Each task is delimited by BEGIN EXPORT and END EXPORT commands</li><li>• The LAYOUT command appears before any IMPORT command that references it</li></ul>

# LOGDATA

## Purpose

Supplies parameters to the LOGMECH command beyond those needed by the logon mechanism, such as user ID and password, to successfully authenticate the user. The LOGDATA command is optional. Whether or not parameters are supplied and the values and types of parameters depend on the selected logon method.

LOGDATA is only available on network-based platforms.

## Syntax

```
.LOGDATA logdata_string ;
```

2409A054

where

Syntax Element	Description
<i>logdata_string</i> ' <i>logdata_string</i> '	Parameters required for the logon mechanism specified using “LOGMECH” on page 108  For information about the logon parameters for supported mechanisms, see the <i>Security Administration</i> guide.  The string is limited to 64 KB and must be in the session character set. To specify a string containing white space or other special characters, enclose the data string in single quotes.

## Usage Notes

For more information about logon security, see *Security Administration*.

## Example

If used, the LOGDATA and LOGMECH commands must precede the LOGON command. The commands themselves may occur in any order.

The following example demonstrates using the LOGDATA, LOGMECH, and LOGON commands in combination to specify the Kerberos logon authentication method and associated parameters:

```
.logmech KRB5
.logdata joe@domain1@mypassword
.logon cs4400s3
```

---

# LOGMECH

## Purpose

Identifies the appropriate logon mechanism by name. If the mechanism specified requires parameters other than user ID and password for authentication, the LOGDATA command provides these parameters. The LOGMECH command is optional and available only on network-attached systems.

## Syntax

```
.LOGMECH _____ logmech_name _____ ;
```

2409A053

where

Syntax Element	Description
<i>logmech_name</i>	Defines the logon mechanism. For a discussion of supported logon mechanisms, see <i>Security Administration</i> . The name is limited to 8 bytes; it is not case-sensitive.

## Usage Notes

Every session to be connected requires a mechanism name. If none is supplied, a default mechanism can be used instead, as defined on either the server or client system in an XML-based configuration file.

For more information about logon security, see *Security Administration*.

## Example

If used, the LOGDATA and LOGMECH commands must precede the LOGON command. The commands themselves may occur in any order.

The following 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 cs4400s3
```

# LOGOFF

## Purpose

The LOGOFF command disconnects all active sessions from the Teradata Database and terminates FastExport.

## Syntax



where

Syntax Element	Description
<i>retcode</i>	[Optional] Completion code to be returned to the client operating system If a <i>retcode</i> is not specified, FastExport returns the appropriate terminating return code.

## Usage Notes

[Table 39](#) describes the things to consider when using the LOGOFF command.

Table 39: LOGOFF Command Usage Notes

Topic	Usage Notes
Optional Completion Code	<p>The optional completion code value, <i>retcode</i>, can be specified as a conditional or an arithmetic expression, evaluated to a single integer.</p> <p>The LOGOFF command processes whenever the highest return code reached was no more than 04 (warning). Any return code other than 00 or 04 terminates the FastExport job.</p> <p>If a serious error terminates the program before the LOGOFF command is processed, the return code output is the value generated by the error condition rather than the <i>retcode</i> value specified as a LOGOFF command option.</p>

Table 39: LOGOFF Command Usage Notes (continued)

Topic	Usage Notes
Terminating Return Codes	<p>When a FastExport job terminates, and an optional <i>retcode</i> value is not specified, the utility returns a code indicating the way the job completed:</p> <ul style="list-style-type: none"> <li>• Code 0—job completed normally</li> <li>• Code 4—a warning condition occurred. Warning conditions do not terminate the job.</li> <li>• Code 8—a user error, such as a syntax error in the FastExport job script, terminated the job</li> </ul> <p><b>Note:</b> The following Teradata Database error messages produce a return code of 08: 3600                    3692                    3695</p> <ul style="list-style-type: none"> <li>• Code 12—a fatal error terminated the job. A fatal error is any error other than a user error.</li> <li>• Code 16—no message destination is available</li> </ul> <p>For a complete description of Teradata Database error messages, refer to the <i>Messages</i> reference documentation.</p>
When Permitted	The LOGOFF command is permitted at any point in the input script. It logs off immediately.
Automatic Logoff	<p>FastExport performs an automatic logoff function if:</p> <ul style="list-style-type: none"> <li>• All of the export data from the Teradata Database has been processed without encountering a LOGOFF command</li> <li>• The program fails because of an error</li> </ul>

## Example

The following example uses a logical expression as the *retcode* specification:

```
.LOGOFF &SYSRC > 8
```

If the expression is true, the *retcode* is 1. If false, it is 0.

# LOGON

## Purpose

The LOGON command establishes a Teradata SQL session with the Teradata Database.

The ACCEPT and SET commands are valid commands preceding LOGON and LOGTABLE commands.

## Syntax

### Standard LOGON Syntax

```
.LOGON tdpid / username , password 'acctid' ;
```

2409A034

**Note:** On the z/OS or z/VM platform, with the use of the User Logon Exit routine in TDP, the user name is not required. See *Teradata Director Program Reference* for more information.

### Single Sign-on LOGON Syntax

```
.LOGON tdpid / username , password 'acctid' ;
```

2410A005

**Note:** On the Windows platform, if logon encryption is enabled on the gateway, then single sign-on is disabled on the client and standard logon syntax should be used instead

where

Syntax Element	Description
<i>acctid</i>	Account identifier associated with the <i>username</i> An account identifier can have up to 30 bytes. If an <i>acctid</i> is not specified, FastExport uses the default identifier defined when the user was created.
<i>password</i>	Password associated with the <i>username</i> A password can have up to 30 bytes.

Syntax Element	Description
<i>tdpid</i>	Optional character string that identifies the name of a TDP If the <i>tdpid</i> is not specified, FastExport uses the default TDP established by the system administrator. <b>Note:</b> For channel-attached systems, the <i>tdpid</i> string must be in the form: TDP <i>n</i> where <i>n</i> is the TDP identifier.
<i>username</i>	User identifier of up to 30 bytes

**Note:** The period preceding the LOGON command is optional.

## Usage Notes

[Table 40](#) describes the things to consider when using the LOGON command.

Table 40: LOGON Command Usage Notes

Topic	Usage Notes
Command Frequency and Placement	A LOGON command is required for each invocation of the FastExport utility. One LOGON command is allowed for each invocation of the FastExport utility, and it must precede any other FastExport commands except RUN FILE command and LOGTABLE command.



Table 40: LOGON Command Usage Notes (continued)

Topic	Usage Notes
Logon Parameters	<p>For standard LOGON, the parameters (<i>tdpid</i>, <i>username</i>, <i>password</i>, and <i>acctid</i>) are used in all sessions established with the Teradata Database. The LOGON command may occur only once.</p> <p>For single sign-on LOGON, if the Gateway to Teradata Database is configured to use single sign-on (SSO), and the Teradata client machine has already been logged on, the machine name, user name, and password are not required in the LOGON command. The user name and password combination specified when the Teradata client machine was logged on are authenticated via network security for a SSO such that valid Teradata users will be permitted to log on to the Teradata Database. The use of SSO is strictly optional, unless the Gateway has been configured to accept only SSO-style logons.</p> <p>To connect to a Teradata Database other than the one currently logged on, the <i>TDPid</i> must be included in the LOGON command. If the <i>TDPid</i> is not specified, the default contained in <i>clispb.dat</i> will be used. (For information about setting defaults, see the <i>Teradata Call-Level Interface Version 2 Reference for Network-Attached Systems</i> book.)</p> <p>To be interpreted correctly, the <i>TDPid</i> must be followed by the slash separator (<i>/</i>), to distinguish the <i>TDPid</i> from a Teradata Database username. For example, to connect to <i>slugger</i>, enter one of the following:</p> <pre>.LOGON slugger/;</pre> <pre>.LOGON slugger/,, 'acctinfo';</pre> <p>If an account ID is to be used, the optional account ID must be specified in the LOGON command.</p>
Using LOGON With the LOGTABLE Command	<p>Both the LOGON and LOGTABLE commands are required.</p> <p>LOGON and commands may appear in any order, but must precede other commands except RUN FILE commands used to identify the file containing the LOGON command.</p> <p>If the LOGON command is entered first, FastExport warns that the LOGTABLE command is also required.</p>

## Example

The following example presents both the LOGON and LOGTABLE commands as they typically occur:

```
.logtable logtable001;
.logon tdpX/me,paswd;
```

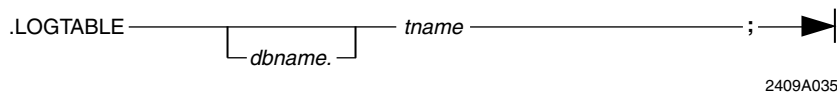
# LOGTABLE

## Purpose

The LOGTABLE command specifies a restart log table for the FastExport checkpoint information. FastExport uses the information in the restart log table to restart jobs that are halted because of a Teradata Database or client system failure.

The ACCEPT and SET commands are valid commands preceding LOGON and LOGTABLE commands.

## Syntax



where

Syntax Element	Description
<i>dbname</i>	Name of the database under which the log table exists  The default is the database name associated with the username specified in the LOGON command. FastExport searches for or creates the table ( <i>tname</i> ) in that database unless another database name is specified in this option.
<i>tname</i>	Name of the restart log table

## Usage Notes

[Table 41](#) describes the things to consider when using the LOGTABLE command.

Table 41: LOGTABLE Command Usage Notes

Topic	Usage Notes
Using LOGTABLE with the LOGON command	Both the LOGTABLE and LOGON commands are required.  LOGTABLE and LOGON commands may appear in any order, but must precede other commands except RUN commands used to identify the file containing the LOGON command.  If the LOGON command is entered first, FastExport warns that the LOGTABLE command is also required.

Table 41: LOGTABLE Command Usage Notes (continued)

Topic	Usage Notes
The Restart Log Table	<p>The table specified as the FastExport restart log table does not have to be fully qualified.</p> <p><b>Note:</b> It is critical that the restart log table not be shared between two or more FastExport jobs. Each FastExport job must have its own restart log table, to ensure proper operation. Failure to use a distinct log table for each FastExport job will cause unexpected results.</p>
Specifying a New or Existing Table	<p>If a table is specified that does not exist, FastExport creates the table and uses it as the restart log during this invocation of the utility.</p> <p>If a table is specified that already exists, then FastExport checks the table to determine whether the current invocation of the utility is a restart operation.</p>
Maintaining the Restart Log Table	<p>FastExport automatically maintains the restart log table. If the table is manipulated in any way, it will invalidate the restart capability.</p> <p>The only valid user maintenance function is to drop the restart log table—never delete rows from the table.</p>
Changing the <i>dbname</i> Specification	<p>The LOGTABLE <i>dbname</i> option must be used to change the <i>dbname</i> specification for a FastExport operation. A subsequent Teradata SQL DATABASE statement, which must appear after the LOGTABLE commands and LOGON commands, cannot be used to change the <i>dbname</i> specification.</p>
Required Access Privilege	<p>The following privileges on the database containing the specified restart log table are required:</p> <ul style="list-style-type: none"> <li>• CREATE TABLE</li> <li>• INSERT</li> <li>• UPDATE</li> <li>• SELECT</li> </ul>

## Example

The following example presents both the LOGTABLE command and the LOGON command as they typically occur:

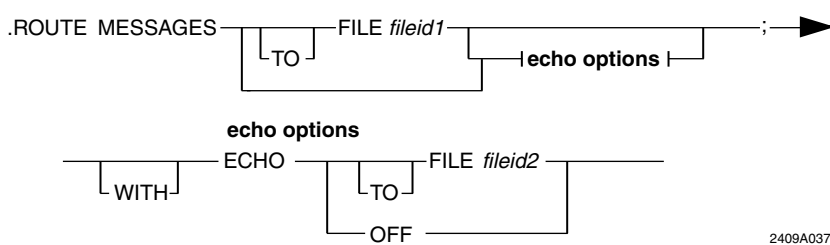
```
.logtable Mine.Logtable001;
.logon tdpx/me,paswd;
```

# ROUTE MESSAGES

## Purpose

The ROUTE MESSAGES command identifies an alternate destination for the report output produced by the FastExport utility. One or more ROUTE MESSAGES command may be included anywhere in the command stream.

## Syntax



2409A037

where

Syntax Element	Description
ECHO	<p>Additional destination, with a <i>fileid</i> specification</p> <p>For example, use the ECHO keyword to specify that messages be captured in a file (<i>fileid2</i>) while still being written to the terminal.</p> <p><b>Note:</b> The ECHO OFF specification cancels the additional file specification of a previously established ECHO destination.</p>
<i>fileid1</i> and <i>fileid2</i>	<p>Alternate message destinations in the external system:</p> <ul style="list-style-type: none"> <li>In z/OS, the <i>fileid</i> is a DDNAME. (See the “z/OS <i>fileid</i> Usage Rules” topic in the “Usage Notes” subsection.)</li> <li>In UNIX and Windows, the <i>fileid</i> is the path name for a file If the path name has embedded white space characters, must enclose the entire path name in single or double quotes.</li> <li>In z/VM, the <i>fileid</i> is a FILEDEF name</li> </ul> <p>If the same destination with both <i>fileid1</i> and <i>fileid2</i> parameters is specified, FastExport duplicates the messages at each destination.</p>

## Usage Notes

Table 42 describes the things to consider when using the ROUTE MESSAGES command.

Table 42: ROUTE MESSAGES Command Usage Notes

Topic	Usage Notes
Specifying the System Console/Standard Output Device	<p>Use the asterisk (*) character as the <i>fileid1</i> or <i>fileid2</i> specifications to route messages to the system console/standard output (stdout) device.</p> <p>The system console is the:</p> <ul style="list-style-type: none"> <li>• Display screen in interactive mode</li> <li>• Standard output device in batch mode</li> </ul> <p>For more information about the display screen and standard output devices, see “File Requirements” on page 21.</p>
Default Message Destinations	<p>If the ROUTE MESSAGES command is not used, FastExport writes output messages to:</p> <ul style="list-style-type: none"> <li>• DDNAME SYSPRINT in z/VM and z/OS</li> <li>• stdout in UNIX and Windows</li> </ul>
z/OS <i>fileid</i> Usage Rules	<p>If a DDNAME is specified, FastExport writes messages to the specified source.</p> <p>A DDNAME must obey the same construction rules as Teradata SQL column names except that:</p> <ul style="list-style-type: none"> <li>• The “at” character (@) is allowed as an alphabetic character</li> <li>• The underscore character (_) is not allowed</li> </ul> <p>The DDNAME must obey the applicable rules of the external system.</p> <p>If the DDNAME represents a data source on magnetic tape, the tape may be either labeled or nonlabeled, as supported by the operating system.</p>

## Example

In the following example, the messages are written to the file designated by OUTPUT from this point unless redirected by another ROUTE MESSAGES command:

```
.ROUTE MESSAGES FILE OUTPUT;
```

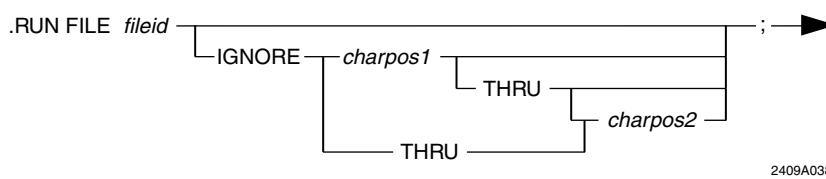
**Note:** On UNIX and Windows platforms, if the same *outfilename* is used to redirect *stdout* and as the *fileid* in a ROUTE MESSAGES WITH ECHO command, the results written to *outfilename* may be incomplete due to conflicting writes to the same file.

# RUN FILE

## Purpose

The RUN FILE command invokes the specified external file as the current source for utility commands and statements.

## Syntax



2409A038

where

Syntax Element	Description
<i>fileid</i>	Data source of the external system The external system DD (or similar) statement specifies a file: <ul style="list-style-type: none"> <li>In UNIX and Windows, the <i>fileid</i> is the path name for a file. If the path name has embedded white space characters, enclose the entire path name in single or double quotes.</li> <li>In z/VM, the <i>fileid</i> is a FILEDEF name</li> <li>In z/OS, a DDNAME. (See the “z/OS <i>fileid</i> Usage Rules” topic in the “Usage Notes” subsection.)</li> </ul>
IGNORE <i>charpos1</i> and <i>charpos2</i>	Start and end character positions of a field in each input record that contains extraneous information. If one of the following is specified: <ul style="list-style-type: none"> <li><i>charpos1</i>, then FastExport ignores only the single specified character</li> <li><i>charpos1</i> THRU, then FastExport ignores all characters from <i>charpos1</i> through the end of the record</li> <li>THRU <i>charpos2</i>, then FastExport ignores all characters from the beginning of the record through <i>charpos2</i></li> <li><i>charpos1</i> THRU <i>charpos2</i>, then FastExport ignores all characters from <i>charpos1</i> through <i>charpos2</i></li> </ul>

## Usage Notes

Table 43 describes the things to consider when using the RUN FILE command.

Table 43: RUN FILE Command Usage Notes

Topic	Usage Notes
Specifying the System Console/Standard Input Device	<p>Use the asterisk (*) character as the <i>fileid</i> specification for the system console/standard input (<i>stdin</i>) device.</p> <p>The system console is the:</p> <ul style="list-style-type: none"> <li>• Keyboard in interactive mode</li> <li>• Standard input device in batch mode</li> </ul> <p>For more information about the keyboard and standard input devices, see “File Requirements” on page 21.</p>
z/OS <i>fileid</i> Usage Rules	<p>If a DDNAME is specified, FastExport reads data records from the specified source.</p> <p>A DDNAME must obey the same construction rules as Teradata SQL column names except that:</p> <ul style="list-style-type: none"> <li>• The “at” character (@) is allowed as an alphabetic character</li> <li>• The underscore character (_) is not allowed</li> </ul> <p>The DDNAME must obey the applicable rules of the external system.</p> <p>If the DDNAME represents a data source on magnetic tape, the tape may be either labeled or nonlabeled, as supported by the operating system.</p>
Executing the RUN FILE Command	<p>After FastExport executes the RUN FILE command, it reads additional commands from the specified source until a LOGOFF command or end-of-file condition is encountered, whichever occurs first.</p> <p>An end-of-file condition automatically causes FastExport to resume reading its commands and DML statements from the previously active source:</p> <ul style="list-style-type: none"> <li>• SYSIN for z/VM and z/OS</li> <li>• stdin (normal or redirected) for UNIX and Windows</li> </ul> <p><b>Note:</b> SYSIN/stdin remains the active input source after FastExport processes any user-provided invocation parameters.</p>
Nested RUN Commands	<p>The source specified by a RUN FILE command can have up to 16 levels of nested RUN commands.</p>

# SET

## Purpose

The **SET** command assigns a data type and a value to a FastExport utility variable. The SET command is a valid command preceding LOGON and LOGTABLE commands.

## Syntax

```
.SET var [TO] expression ;
```

2409A039

where

Syntax Element	Description
<i>var</i>	Name of the FastExport utility variable to be set to the evaluated <i>expression</i>

## Usage Notes

[Table 44](#) describes the things to consider when using the SET command.

Table 44: SET Command Usage Notes

Topic	Usage Notes
Declaring Variables	Variables need not be declared in advance to be the object of the SET command. If a variable does not already exist, FastExport creates it. Variables used to the right of TO in the expression must be declared in advance.
Changing the Data Type	The SET command also dynamically changes the data type to that of the assigned value if it had already been defined. If the expression evaluates to a numeric value, the symbol is assigned an integer value, as in: .SET FOONUM TO -151 ; If the expression is a quoted string, the symbol is assigned a string value, as in: .SET FOOCHAR TO '-151' ; The minimum and maximum limits for floating point data types are as follows: 4.0E-75 <=abs(float variable)<7.0E75
Variable Substitution	A FastExport variable can be substituted wherever substitution is allowed.



## Example 1

Boolean operations can be performed using the SET command, such as:

```
.SET A TO 0;  
.SET B TO NOT &A;  
.DISPLAY '&B' TO FILE CONSOLE = = > The value of 1 is returned
```

## Example 2

FastExport supports logical and relational operators using the SET command, such as:

```
.SET A TO 1;  
.SET B TO 0;  
.SET Z TO (A or B);
```

## Example 3

FastExport supports concatenation of variables, using the SET command, such as:

```
.SET C TO 1;  
.SET D TO 2;  
.SET X TO &C.&D;
```

In this example, X evaluates to 12.

If a decimal point is added to the concatenated variables, then X evaluates to 1.2, as in:

```
.SET C TO 1;  
.SET D TO 2;  
.SET X TO &C..&D;
```


---

# SYSTEM

## Purpose

The SYSTEM command submits an operating system command to the client environment during a FastExport operation.

## Syntax

```
.SYSTEM 'oscommand' _____ ; 
```

2409A040

where

Syntax Element	Description
<i>oscommand</i>	Any legal command in the client operating system

## Usage Notes

The SYSTEM command suspends the current FastExport operation to execute the client operating system command.

When the client operating system command completes, FastExport displays the return code from the invoked command and updates the &SYSRC variable.

## Example

The following example deletes the file `/home/fexpuser/tests/out1` if it exists. The command string then creates a new `/home/fexpuser/tests/out1` file to contain the exported records from the SELECT statement.

```
.SYSTEM 'rm -f /home/fexpuser/tests/out1';  
.BEGIN EXPORT;  
SEL * FROM table1;  
.EXPORT OUTFILE /home/fexpuser/tests/out1;  
.END EXPORT;
```

# How to Read Syntax Diagrams

This appendix describes the conventions that apply to reading the syntax diagrams used in this book.

## Syntax Diagram Conventions

### Notation Conventions

Item	Definition / Comments
Letter	An uppercase or lowercase alphabetic character ranging from A through Z.
Number	A digit ranging from 0 through 9. Do not use commas when typing a number with more than 3 digits.
Word	Variables and reserved words. <ul style="list-style-type: none"> <li>UPPERCASE LETTERS represent a keyword. Syntax diagrams show all keywords in uppercase, unless operating system restrictions require them to be in lowercase.</li> <li>lowercase letters represent a keyword that you must type in lowercase, such as a UNIX command.</li> <li><i>lowercase italic letters</i> represent a variable such as a column or table name. Substitute the variable with a proper value.</li> <li><b>lowercase bold letters</b> represent a variable that is defined immediately following the diagram that contains the variable.</li> <li><u>UNDERLINED LETTERS</u> represent the default value. This applies to both uppercase and lowercase words.</li> </ul>
Spaces	Use one space between items such as keywords or variables.
Punctuation	Type all punctuation exactly as it appears in the diagram.

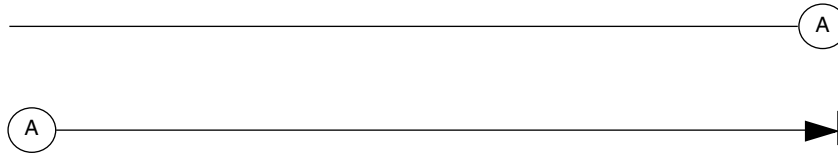
### Paths

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:



FE0CA002

When you see a circled letter in a syntax diagram, go to the corresponding circled letter and continue reading.

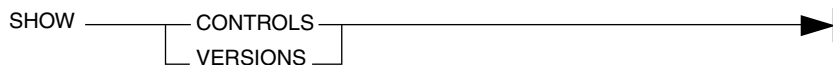
## Required Entries

Required entries appear on the main path:



FE0CA003

If you can choose from more than one entry, the choices appear vertically, in a stack. The first entry appears on the main path:



FE0CA005

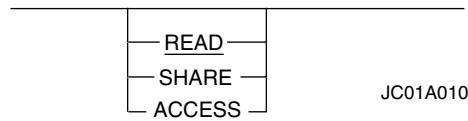
## Optional Entries

You may choose to include or disregard optional entries. Optional entries appear below the main path:



FE0CA004

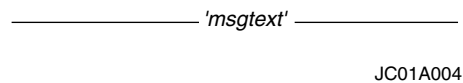
If you can optionally choose from more than one entry, all the choices appear below the main path:



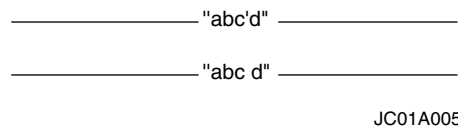
Some commands and statements treat one of the optional choices as a default value. This value is UNDERLINED. It is presumed to be selected if you type the command or statement without specifying one of the options.

## Strings

Strings appear in single quotes:



If the string text includes a single quote or a blank space, the string appears in double quotes:



## 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.

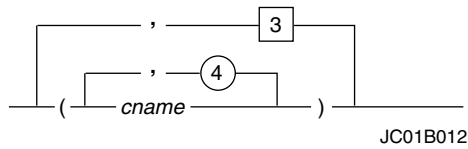


In the above syntax, the following formats are valid:

- 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:



Read loops from right to left.

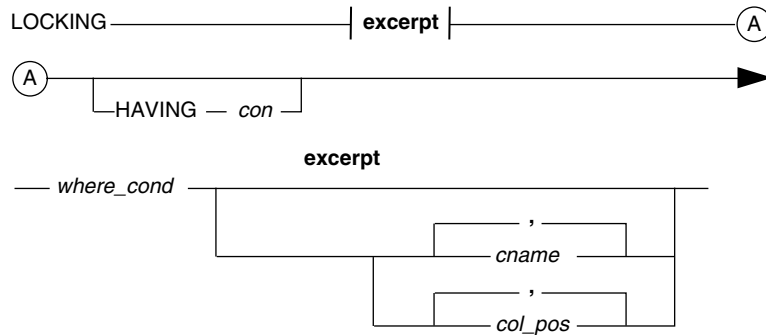
The following conventions apply to loops:

Loop Convention	Description
A maximum number of entries is allowed.	The number appears in a circle on the return path. In the example, you may type <i>cname</i> a maximum of 4 times.
A minimum number of entries is required.	The number appears in a square on the return path. In the example, you must type at least three groups of column names.
A separator character is required between entries.	The character appears on the return path. If the diagram does not show a separator character, use one blank space. In the example, the separator character is a comma.
A delimiter character is required around entries.	The beginning and end characters appear outside the return path. Generally, a space is not needed between delimiter characters and entries. In the example, the delimiter characters are the left and right parentheses.

## 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.

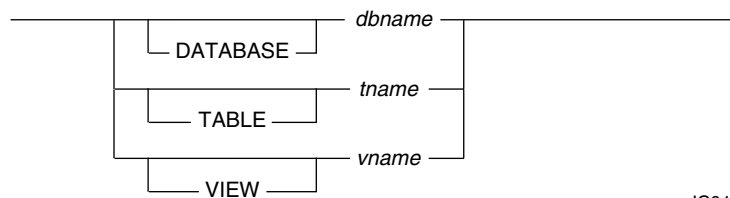
The boldface excerpt name and the excerpted phrase appears immediately after the main diagram. The excerpted phrase starts and ends with a plain horizontal line:



JC01A014

## Multiple Legitimate Phrases

In a syntax diagram, it is possible for any number of phrases to be legitimate:

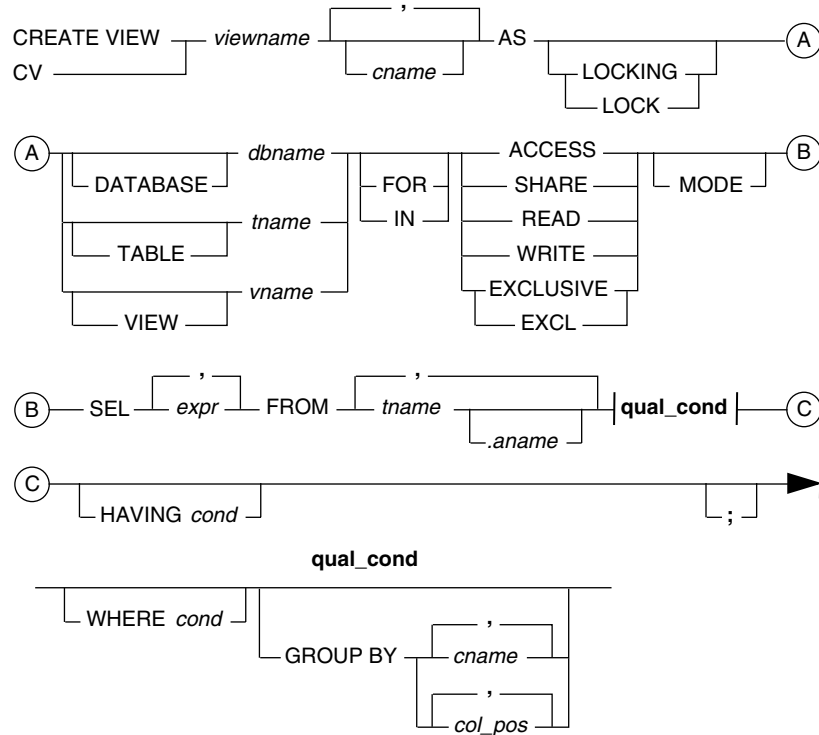


JC01A016

In this example, any of the following phrases are legitimate:

- *dbname*
- *DATABASE dbname*
- *tname*
- *TABLE tname*
- *vname*
- *VIEW vname*

## Sample Syntax Diagram



JC01A018

## 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.



# Invocation Examples

This appendix provides JCL and command examples for invoking FastExport on z/VM, z/OS, and on UNIX and Windows systems.

## z/VM

This section provides JCL and command examples for invoking FastExport on z/VM systems.

### Reduced Print Output (BRIEF) Parameter

#### Sample Command:

```
fastexpt stress1 brief
```

**Note:** *stress1* is a sample input script.

#### Sample Output:

```
**** 10:15:36 UTY2414 BRIEF option is enabled.
=====
      FastExport Utility      Release FEXP.13.00.00.000      =
      Platform VM              =
=====
      Copyright 1990-2008, Teradata Corporation. ALL RIGHTS RESERVED. =
=====
**** 10:15:36 UTY2411 Processing start date: FRI MAY 18,
=====
      Logon/Connection              =
=====
0001 .logtable fexp_test;
0002 .logon tdp9/fexp,;
**** 10:15:37 UTY8400 Default character set: EBCDIC
**** 10:15:52 UTY6211 A successful connect was made to the DBS.
**** 10:15:52 UTY6217 Logtable 'fexp.FEXP_TEST' has been created.
.
.
.
```

### Character Set Selection (CHARSET) Parameter

#### Sample Command:

```
fastexpt stress1 charset=ebcdic
```

**Note:** *stress1* is a sample input script.

### Sample Output:

```
**** 10:27:56 UTY2407 Run time parameters in effect are:
CHARSET=EBCDIC.
=====
=
=   FastExport Utility      Release FEXP.13.00.00.000      =
=   Platform z/VM =
=
=====
=
=   Copyright 1990-2008, Teradata Corporation. ALL RIGHTS RESERVED. =
=
=====
**** 10:27:56 UTY2411 Processing start date: FRI MAY 18,
=====
=
=   Logon/Connection
=
=====

0001 .logtable fexp_test;
0002 .logon tdp9/fexp,;
**** 10:28:10 UTY6211 A successful connect was made to the DBS.
**** 10:28:10 UTY6217 Logtable 'fexp.FEXP_TEST' has been created.
.
.
.
```

## Error Logging (ERRLOG) Parameter

### Sample Command:

```
fastexpt stress1 errlog=foo
```

**Note:** *stress1* is a sample input script. *foo* must be defined. In this example *foo* was defined by:

```
filedef foo disk error file a
```

### Sample Output:

```
**** 10:31:54 UTY2413 Error Logging is enabled: FOO
=====
=
=   FastExport Utility      Release FEXP.13.00.00.000      =
=   Platform z/VM          =
=
=====
=
=   Copyright 1990-2008, Teradata Corporation. ALL RIGHTS RESERVED. =
=
=====
**** 10:31:54UTY2411 Processing start date: FRI MAY 18, 2008
=====
=
=   Logon/Connection
=
```

```

=====
0001 .logtable fexp_test;
0002 .logon tdp9/fexp,;
**** 10:32:26 UTY8400 Default character set: EBCDIC
/*****
/*
/* Test handling multiple Fexp tasks.
/*
/*****
.
.
.
=====

```

## Specify Multiple Parameters

### Sample Command:

```
fastexpt stress1 charset=ebcdic brief
```

**Note:** *stress1* is a sample input script.

### Sample Output:

```

**** 10:43:18 UTY2407 Run time parameters in effect are:          CHARSET=EBCDIC.
**** 10:43:18 UTY2414 BRIEF option is enabled.
=====
=      FastExport Utility      Release FEXP.13.00.00.000          =
=      Platform z/VM           =
=====
=
=      Copyright 1990-2008, Teradata Corporation. ALL RIGHTS RESERVED.  =
=
=====
**** 10:43:18 UTY2411 Processing start date: FRI MAY 18, 2008
=====
=      Logon/Connection
=====
0001 .logtable fexp_test;
0002 .logon tdp9/fexp,;
**** 10:43:32 UTY6211 A successful connect was made to the DBS.
**** 10:43:32 UTY6217 Logtable 'fexp.FEXP_TEST' has been created.
0003
/*****
/*
/* Test handling multiple Fexp tasks.
/*
/*****
.
.
.
=====

```

## z/OS

This section provides JCL and command examples for invoking FastExport on z/OS systems.

## Reduced Print Output (BRIEF) Parameter

### Sample JCL:

```
//FEXPMLD05 JOB (78030000),'FOO',  
//          REGION=4096K  
//JOBLIB   DD DISP=SHR,DSN=TERADATA.APLOAD  
//          DD DISP=SHR,DSN=TERADATA.TRLOAD  
//FEXPRUN  EXEC PGM=XPORT,PARM='BRIEF'  
//SYSPRINT DD SYSOUT=A  
//SYSIN    DD DATA,DLM=##  
.LOGTABLE FEXP_TABLE5;  
.LOGON TDP9/FEXP,FEXP;  
.VERSION;  
.LOGOFF;  
##
```

### Sample Output:

```
000081 *****  
000082  
000083 **** 14:23:13 UTY2414 BRIEF option is enabled.  
000084 =====  
000085 =   FastExport Utility      Release FEXP.13.00.00.000   =  
000086 =   Platform z/OS =  
000087 =====  
000088 =   =  
000099 =   Copyright 1990-2008, Teradata Corporation. ALL RIGHTS RESERVED. =  
000090 =   =  
000091 =====  
000092 **** 14:23:13 UTY2411 Processing start date: FRI MAY 18, 2008  
000093 =====  
000094 =           Logon/Connection           =  
000095 =====  
000096 0001 .LOGTABLE FEXP_TABLE5;  
000097 0002 .LOGON TDP9/FEXP,;  
000098 **** 14:23:16 UTY8400 Default character set: EBCDIC  
000099 **** 14:23:46 UTY6211 A successful connect was made to the DBS.  
000100 **** 14:23:46 UTY6217 Logtable 'FEXP.FEXP_TABLE5' has been created.  
000101  
000102 0003 .VERSION;  
.  
.  
.
```

## Character Set Selection (CHARSET) Parameter

### Sample JCL:

```
//FEXPMLD15 JOB (78030000),'FOO',  
//          REGION=4096K  
//JOBLIB   DD DISP=SHR,DSN=TERADATA.APLOAD  
//          DD DISP=SHR,DSN=TERADATA.TRLOAD  
//FEXPRUN  EXEC PGM=XPORT,PARM='CHARSET=EBCDIC'  
//SYSPRINT DD SYSOUT=A  
//SYSIN    DD DATA,DLM=##  
.LOGTABLE FEXP_TABLE5;  
.LOGON TDP9/FEXP,FEXP;
```

```
.VERSION;
.LOGOFF;
##
```

### Sample Output:

```
000082
000083 **** 15:43:31 UTY2407 Run time parameters in effect are:
      CHARSET=EBCDIC.
000084 =====
000085 =
000086 =      FastExport Utility      Release FEXP.13.00.00.000      =
000087 =      Platform z/OS      =
000088 =
000089 =====
000090 =
000091 =      Copyright 1990-2008, Teradata Corporation. ALL RIGHTS RESERVED.=
000092 =
000093 =====
000094 **** 15:43:31 UTY2411 Processing start date: FRI MAY 18, 2008
000095 =====
000096 =
000097 =      Logon/Connection
000098 =
000099 =====
000100 0001 .LOGTABLE FEXP_TABLE5;
000101 0002 .LOGON TDP9/FEXP,;
000102 **** 15:43:52 UTY6211 A successful connect was made to the DBS.
000103 **** 15:43:52 UTY6217 Logtable 'FEXP.FEXP_TABLE5' has been created.
000104 =====
000105 =
000106 =      Processing Control Statements
000107 =
000108 =====
.
.
.
```

## Error Logging (ERRLOG) Parameter

### Sample JCL:

```
//FEXPMLD16 JOB (78030000),'FOO',
//          REGION=4096K
//JOBLIB   DD  DISP=SHR,DSN=TERADATA.APPLOAD
//          DD  DISP=SHR,DSN=TERADATA.TRLOAD
//FEXPRUN  EXEC PGM=XPORT,PARM='ERRLOG=FOO'
//FOO      DD  DSN=FEXP.FOO.OUTPUT,DISP=SHR
//SYSPRINT DD  SYSOUT=A
//SYSIN    DD  DATA,DLM=##
.LOGTABLE FEXP_TABLE5;
.LOGON TDP9/FEXP,FEXP;
.VERSION;
.LOGOFF;
##
```

### Sample Output:

```

000085 *****
000086
000087 **** 15:19:47 UTY2413 Error Logging is enabled: FOO
000088 =====
000089 =
000090 = FastExport Utility      Release FEXP.13.00.00.000      =
000091 = Platform z/OS           =
000092 =
000093 =====
000094 =
000095 = Copyright 1990-2008, Teradata Corporation. ALL RIGHTS RESERVED. =
000096 =
000097 =====
000098 **** 15:19:47 UTY2411 Processing start date: FRI MAY 18, 2008
000099 =====
000100 =
000101 = Logon/Connection
000102 =
000103 =====
000104 0001 .LOGTABLE FEXP_TABLE5;
000105 0002 .LOGON TDP9/FEXP,;
000106 **** 15:19:48 UTY8400 Default character set: EBCDIC
000107 **** 15:20:12 UTY6211 A successful connect was made to the DBS.
000108 **** 15:20:12 UTY6217 Logtable 'FEXP.FEXP_TABLE5' has been created.
000109 =====
000110 =
000111 = Processing Control Statements
000112 =
000113 =====
.
.
.

```

## Specify Multiple Parameters

### Sample JCL:

```

//FEXPMLD17 JOB (78030000),'FOO',
//          REGION=4096K
//JOBLIB   DD DISP=SHR,DSN=TERADATA.APPLOAD
//          DD DISP=SHR,DSN=TERADATA.TRLOAD
//FEXP_RUN EXEC PGM=XPORT,PARM='BRIEF,CHARSET=EBCDIC'
//SYSPRINT DD SYSOUT=A
//SYSIN   DD DATA,DLM=##
.LOGTABLE FEXP_TABLE5;
.LOGON TDP9/FEXP,FEXP;
.VERSION;
.LOGOFF;
##

```

### Sample Output:

```

000083 **** 15:49:40 UTY2414 BRIEF option is enabled.
000084 **** 15:49:40 UTY2407 Run time parameters in effect are:
CHARSET=EBCDIC.
000085 =====
000086 = FastExport Utility      Release FEXP.13.00.00.000      =
000087 = Platform z/OS           =

```

```

00088 =====
00089 =
00090 =          Copyright 1990-2008, Teradata Corporation. ALL RIGHTS RESERVED. =
00091 =
00092 =====
00093 **** 15:49:40 UTY2411 Processing start date: FRI MAY 18, 2008
00094 =====
00095 =          Logon/Connection
00096 =====
00097 0001 .LOGTABLE FEXP_TABLE5;
00098 0002 .LOGON TDP9/FEXP,;
00099 **** 15:50:10 UTY6211 A successful connect was made to the DBS.
00100 **** 15:50:10 UTY6217 Logtable 'FEXP.FEXP_TABLE5' has been created.
00101
00102 0003 .VERSION;
.
.
.

```

## UNIX and Windows

This section provides JCL and command examples for invoking FastExport on UNIX and Windows systems.

### Run File (-r) Parameter

The typical selection is:

```
fexp -r '.RUN FILE fexp.startup;' < infilename > outfilename
```

#### Sample Command:

```
fexp -r '.run file logon;' < job.fexp
```

#### Sample Output:

```

=====
=
=          FastExport Utility      Release FEXP.13.00.00.000
=          Platform MP-RAS
=
=====
=
=          Copyright 1990-2008, Teradata Corporation. ALL RIGHTS RESERVED. =
=
=====
**** 13:58:25 UTY2411 Processing start date: FRI MAY 18, 2008
=====
=
=          Logon/Connection
=
=====
0002 .logtable fexplog;
0003 .logon cs4300s1/fexp,;
**** 14:01:13 UTY8400 Default character set: ASCII
**** 14:01:17 UTY6211 A successful connect was made to the RDBMS.

```

```
**** 14:01:17 UTY6217 Logtable 'fexp.fexplog' has been created.
=====
=
=           Processing Control Statements           =
=
=====
0004 .begin export sessions 4;
0005 sel * from dbc.sessioninfo;
0006 .export outfile job.out mode record;
0007 .end export;
.
.
.
```

## Reduced Print Output (-b) Parameter

### Sample Command:

```
fexp -b < foo2
```

### Sample Output:

```
**** 15:13:01 UTY2414 BRIEF option is enabled.
=====
=
=           FastExport Utility           Release FEXP.13.00.00.000           =
=           Platform MP-RAS           =
=
=====
=
=           Copyright 1990-2008, Teradata Corporation. ALL RIGHTS RESERVED. =
=
=====
**** 15:13:01 UTY2411 Processing start date: FRI MAY 18, 2008
=====
=
=           Logon/Connection           =
=
=====
0001 .run file logon;
0002 .logon shogun/fexp,;
**** 15:13:01 UTY6214 Reminder: A .Logtable statement must be entered for a
      successful logon.
0003 .logtable fexptest2;
**** 15:13:04 UTY8400 Default character set: KANJIEUC_OU
**** 15:13:08 UTY6211 A successful connect was made to the DBS.
**** 15:13:08 UTY6217 Logtable 'fexp.fexptest2' has been created.
.
.
.
```

## Character Set Selection (-c) Parameter

### Sample Command:

```
fexp -c ascii
```



### Sample Output:

```

**** 15:27:35 UTY2407 Run time parameters in effect are: ASCII.
=====
=
=      FastExport Utility      Release FEXP.13.00.00.000
=      Platform MP-RAS
=
=====
=
=      Copyright 1990-2008, Teradata Corporation. ALL RIGHTS RESERVED.
=
=====
**** 15:27:35 UTY2411 Processing start date: FRI MAY 18, 2008
=====
=
=      Logon/Connection
=
=====
0001 .run file logon;
0002 .logon shogun/fexp,;
**** 15:27:35 UTY6214 Reminder: A .Logtable statement must be entered for a
successful logon.
=====
=
=      Processing Control Statements
=
=====
0003 .logtable fexptest2;
**** 15:27:39 UTY6211 A successful connect was made to the DBS.
**** 15:27:39 UTY6217 Logtable 'fexp.fexptest2' has been created.
.
.
.

```

### Error Logging (-e) Parameter

#### Sample Command:

```
fexp -e errfile < foo2
```

#### Sample Output:

```

**** 15:33:10 UTY2413 Error Logging is enabled: errfile
=====
=
=      FastExport Utility      Release FEXP.13.00.00.000
=      Platform MP-RAS
=
=====
=
=      Copyright 1990-2008, Teradata Corporation. ALL RIGHTS RESERVED.
=
=====
**** 15:33:10 UTY2411 Processing start date: FRI MAY 18, 2008
=====
=
=      Logon/Connection
=
=====
0001 .run file logon;

```

```
0002 .logon shogun/fexp,;
**** 15:33:10 UTY6214 Reminder: A .Logtable statement must be entered for a
successful logon.
=====
=
= Processing Control Statements =
=
=====
0003 .logtable fexptest2;
.
.
.
```

## Specify Multiple Parameters

### Sample Command:

```
fexp -c ascii -b
```

### Sample Output:

```
**** 15:38:31 UTY2407 Run time parameters in effect are: ASCII.
**** 15:38:31 UTY2414 BRIEF option is enabled.
=====
= FastExport Utility Release FEXP.13.00.00.000 =
= Platform MP-RAS =
=====
= Copyright 1990-2008, Teradata Corporation. ALL RIGHTS RESERVED. =
=====
**** 15:38:31 UTY2411 Processing start date: FRI MAY 18, 2008
=====
= Logon/Connection =
=====
0001 .run file logon;
0002 .logon shogun/fexp,;
**** 15:38:31 UTY6214 Reminder: A .Logtable statement must be entered for a
successful logon.
0003 .logtable fexptest2;
**** 15:38:34 UTY6211 A successful connect was made to the DBS.
**** 15:38:34 UTY6217 Logtable 'fexp.fexptest2' has been created.
.
.
.
```

# INMOD, OUTMOD and Notify Exit Routine Examples

This appendix provides examples of INMOD, OUTMOD, and notify exit routines on the following client platforms:

- [z/VM](#)
- [z/OS](#)
- [UNIX](#)
- [Windows](#)

## Table Format

In each case, for z/VM, z/OS and UNIX client systems, the example presumes that a table named TranLogTable has been created on the Teradata Database as follows:

```
CREATE TABLE TranLogTable
  (TranDate Date,
   Region Char(3),
   CustNo Char(12),
   OrderNo Char(6),
   ProdCode Char(8),
   Quantity Integer,
   Price Integer)
  Unique Primary Index(OrderNo, ProdCode);
```

The Windows examples are presented differently.

## Row Format

In each case, for z/VM, z/OS and UNIX client systems, the columns to be selected are: Region, ProdCode, Quantity and Price.

The examples sort response data by Region and ProdCode and return it in indicator mode format. The response records are therefore defined in the same format as that for a row of the TranLogTable on the Teradata Database, plus the leading indicator byte.

Each response row has 20 bytes, consisting of:

Indicator Flags	1 byte
Region	3 bytes
ProdCode	8 bytes
Quantity	4 bytes
Price	4 bytes

In the OUTMOD examples, the FastExport utility passes each response record to a client procedure called ChkTran that modifies selected transaction records and notifies FastExport to either drop or to write each record to the client data set.

The Windows examples are presented differently.

## z/VM

### Generating a COBOL OUTMOD Routine

The following example uses the name CHKTRAN as the member name to generate an OUTMOD routine on z/VM client systems:

```
IDENTIFICATION DIVISION.  
PROGRAM-ID. DYNAMN.  
AUTHOR. USER.  
INSTALLATION. TERADATA.  
DATE-WRITTEN. 12 AUGUST 1992  
DATE COMPLIED.  
SECURITY. OPEN.  
REMARKS.  
    THIS PROCEDURE IS INVOKED BY THE TERADATA FASTEXPORT  
    UTILITY TO PROCESS THE RESPONSE DATA RETURNED FROM THE  
    SAMPLE SELECT.  THE PROCEDURE EXAMINES EACH RESPONSE  
    RECORD TO DETERMINE IF THE RECORD SHOULD BE WRITTEN  
    TO AN ERROR DATA SET AND THEN EITHER DROPPED OR  
    WRITTEN TO THE STANDARD DATA SET.  ONE ERROR DATA SET  
    CONTAINS RECORDS WITH A NULL REGION CODE.  THE OTHER  
    ERROR DATA SET CONTAINS RECORDS WITH A TOTAL SALES  
    VALUE OF LESS THAN $100.  THESE LATTER RECORDS ARE NOT  
    WRITTEN TO THE STANDARD DATA SET.  
ENVIRONMENT DIVISION.  
CONFIGURATION SECTION.  
SOURCE-COMPUTER. IBM-370.  
OBJECT-COMPUTER. IBM-370.  
INPUT-OUTPUT SECTION.  
FILE-CONTROL.  
    SELECT SALES-DROPPED-FILE ASSIGN TO FILE1OUT.  
    SELECT BAD-REGN-SALES-FILE ASSIGN TO FILE2OUT.  
DATA DIVISION.  
FILE SECTION.  
FD SALES-DROPPED-FILE  
    BLOCK CONTAINS 1 RECORDS  
    LABEL RECORDS STANDARD.  
01 DROPPED-TRANLOG.
```

```

02 INDICATORS      PIC 9.
02 REGN            PIC XXX.
02 PRODUCT        PIC X(8).
02 QTY            PIC S9(8) COMP.
02 PRICE          PIC S9(8) COMP.
FD BAD-REGN-SALES-FILE
BLOCK CONTAINS 1 RECORDS
LABEL RECORDS STANDARD.
01 BAD-REGN-TRANLOG.
02 INDICATORS     PIC 9.
02 REGN           PIC XXX.
02 PRODUCT       PIC X(8).
02 QTY           PIC S9(8) COMP.
02 PRICE        PIC S9(8) COMP.
LINKAGE SECTION.
01 ENTRY-TYPE     PIC S9(5) COMP.
01 STATEMENT-NO  PIC S9(5) COMP.
01 RECORD-SIZE   PIC S9(5) COMP.
01 TRANLOG.
05 INDICATORS    PIC 9.
05 REGN          PIC XXX.
05 PRODUCT      PIC X(8).
05 QTY          PIC S9(8) COMP.
05 PRICE        PIC S9(8) COMP.
01 OUTPUT-LENGTH PIC S9(5) COMP.
01 OUTPUT-AREA  PIC XXXX.
PROCEDURE DIVISION USING
    ENTRY-TYPE, STATEMENT-NO, RECORD-SIZE, TRANLOG,
    OUTPUT-LENGTH, OUTPUT-AREA.
BEGIN.
MAIN.
    IF ENTRY-TYPE = 1 THEN
        OPEN OUTPUT SALES-DROPPED-FILE
        OPEN OUTPUT BAD-REGN-SALES-FILE
        GOBACK.
    IF ENTRY-TYPE = 2 THEN
        CLOSE SALES-DROPPED-FILE
        CLOSE BAD-REGN-SALES-FILE
        GOBACK.
    IF ENTRY-TYPE = 3 THEN
        PERFORM TYPE-3
        GOBACK.
    IF ENTRY-TYPE = 4 THEN
        GOBACK.
    IF ENTRY-TYPE = 5 THEN
        CLOSE SALES-DROPPED-FILE
        OPEN OUTPUT SALES-DROPPED-FILE
        CLOSE BAD-REGN-SALES-FILE
        OPEN OUTPUT BAD-REGN-SALES-FILE
        GOBACK.
    IF ENTRY-TYPE = 6 THEN
        OPEN OUTPUT SALES-DROPPED-FILE
        OPEN OUTPUT BAD-REGN-SALES-FILE
        GOBACK.
    DISPLAY "Invalid entry code = " ENTRY-TYPE.
    GOBACK.
TYPE-3.
IF QTY IN TRANLOG * PRICE IN TRANLOG < 100 THEN
    MOVE 0 TO RECORD-SIZE

```

```

WRITE DROPPED-TRANLOG FROM TRANLOG
ELSE
PERFORM TEST-NULL-REGN.
TEST-NULL-REGN.
IF REGN IN TRANLOG = SPACES
MOVE 999 TO REGN IN TRANLOG
WRITE BAD-REGN-TRANLOG FROM TRANLOG.

```

## Omit the Default Entry Point

Use the OSDECK option of the COBOL compiler to omit the default entry point name when generating a dynamically loadable COBOL procedure. If the COBOL source in the preceding example is in file CHKTRAN COBOL, the following z/VM command compiles the source and creates the object module as file CHKTRAN TEXT:

```
COBOL CHKTRAN (OSDECK
```

## Create the Load Module

Use the following commands to create the load module:

```

FILEDEF SYSLIB DISK COBLIBVS TXTLIB *
LKED CHKTRAN (LIBE DYNAMC NAME CHKTRAN AMODE 24 RMODE 24

```

**Note:** The appropriate file mode to the FILEDEF command defining the TXTLIB containing the COBOL modules may have to be added. CHKTRAN LKEDIT and DYNAMC LOADLIB are the two files output from the linkedit.

## Generate a C OUTMOD Routine

The following C example generates an OUTMOD routine on z/VM client systems:

```

/* TITLE CHKTRAN ... Output procedure for TranLogTable export run */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*
/* Purpose This procedure is called by the Teradata FastExport */
/* utility for each response row returned to the host. */
/* The procedure examines each row to determine if the */
/* row should be output to an error data set and then */
/* either dropped or written to the standard data set. */
/* One error data sets contain records with a null */
/* region code the other contains records with a total */
/* sales value less than $100. These latter records */
/* are not written to the standard data set. */
/*
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#include <stdio.h>
#define chkbit(a,b) (((0x80 >> (b-1)) & (a)) ? 1 : 0)
/* Define the structure of the response row */
struct tranlog
{
char Indicators;
char Region[3];
char Product[8];
long Qty;

```

```

    long      Price;
} ;
#define minsale      100      /* minimum sales to report */
FILE *file1, *file2;
int  _dynamn(EntryType, StmtNo,
            RespLen, RespRec,
            OutLen, OutRec)
int      *EntryType;
int      *StmtNo;
int      *RespLen;
struct tranlog*RespRec;
int      *OutLen;
char     *OutRec;
{
long recsize;
struct tranlog recout;
/* case on entry type
*/
switch (*EntryType) {
case 1:
    /* Normal start */
    file1 = fopen("ddn:file1out", "wb");
    file2 = fopen("ddn:file2out", "wb");
    break;
case 2:
    /* EOF for response data */
    fclose(file1);
    fclose(file2);
    break;
case 3:
    /* Process response record */
    if ((RespRec->Qty * RespRec->Price) < minsale)
        {
            *RespLen = 0;
            recout = *RespRec;
            recsize = fwrite(&recout, sizeof(recout), 1, file1);
        }
    else
        {
            /* If the region is null then change it to a dummy region.
            */ if (chkbit(RespRec->Indicators,1))
                {
                    RespRec->Region[1] = '9';
                    RespRec->Region[2] = '9';
                    RespRec->Region[3] = '9';
                    recout = *RespRec;
                    recsize = fwrite(&recout, sizeof(recout), 1, file2);
                }
        }
    break;
case 4:
    /* no checkpoints to worry about */
    break;
case 5:
    /* DBC restart - close and reopen the output files */
    file1 = freopen("ddn:file1out", "wb", file1);
    file2 = freopen("ddn:file2out", "wb", file2);
    break;
case 6:

```

```
/* Host restart same as normal since there are no checkpoints */
file1 = fopen("ddn:file1out", "wb");
file2 = fopen("ddn:file2out", "wb");
break;
default:
    printf("Invalid entry code = %d\n", *EntryType);
    break;
}
return(0);
}
```

## Compile the OUTMOD Routine

Use the SAS LC370 exec to compile the OUTMOD routine. The macro library LC370 MACLIB for use by the compiler must also be established. Use the following commands to compile the OUTMOD routine, assuming that the source is in file CHKTRAN C:

```
GLOBAL MACLIB LC370
LC370 CHKTRAN
```

## Link the Object File

The object file output by the compiler resides in file CHKTRAN TEXT. Link edit the file as a member of the loadlib named DYNAMC. Use the CLINK EXEC, provided by SAS, for this purpose. The following CLINK command assumes that the object module is in file CHKTRAN TEXT:

```
GLOBAL TXTLIB LC370STD LC370BAS
CLINK CHKTRAN (LKED LIBE DYNAMC NAME CHKTRAN AMODE 24
              RMODE 24
```

CHKTRAN LKEDIT and DYNAMC LOADLIB are the two files output from the linkedit.

## Execute the OUTMOD Routine

Before executing the OUTMOD, the run-time libraries must be established and the filedefs are generated for the data input and output to the sample export run.

Because the FastExport utility was written using SAS/C, make the latest version of the SAS/C run time libraries available for execution.

Execution of the FastExport utility also requires the latest version of the Teradata run-time libraries.

In this example, an input record that contains a begin and end date defines the time period for the data to be exported. These dates are in the first eight bytes of the record in year, month, and day integer format. The record is read from the data set defined by DDName InName.

FastExport uses the Teradata Database table Tranlog as the restart log. The logon string is read from the data set defined by DDName PwdName. FastExport-generated response data is written through DDName OutName.

## FastExport Utility Directives

Following are the directives for the FastExport utility:



```
.LOGTABLE TranLog ;          /* define restart log          */
.RUN FILE PwdName ;         /* Get the logon string        */
                             /* from a file                  */
.BEGIN EXPORT               /* Specify export function     */
  SESSIONS 20;              /* number of sessions to be used */
.LAYOUT Control ;          /* Define the control record    */
  .FIELD Date1 * Date ;
  .FIELD Date2 * Date ;
.IMPORT INFILE InName      /* Identify input control file  */
  LAYOUT Control ;        /* and where the description is. */
.EXPORT OUTFILE OutName   /* identify the destination     */
  OUTMOD ChkTran ;       /* file and the procedure       */
                             /* to receive the records.      */
SELECTRegion,              /* provide the SQL SELECT      */
      ProdCode,
      Quantity,
      Price
      From TranLogTable
      WHERE TranDate BETWEEN :Date1 and :Date2
      ORDER BY Region, ProdCode;
.END EXPORT ;              /* terminate the export        */
                             /* operation                    */
.LOGOFF ;                  /* disconnect from the DBC     */
```

The FastExport utility reads the initial commands through stream SYSIN. The default output is through stream SYSPRINT.

The sample OUTMOD procedure writes data to streams FILE1OUT and FILE2OUT. It also uses the Printf call in C or the DISPLAY verb in COBOL which uses stream SYSOUT.

## z/VM Directives

Following is an example of the z/VM directives used to execute the sample FastExport scripts:

```
GLOBAL LOADLIB LSCRTL
GLOBAL TXTLIB CLI
FILEDEF SYSIN DISK CHKTRAN MX$INPUT
FILEDEF PWDNAME DISK PSWD NAME
FILEDEF INNAME DISK TRANIN DATA
FILEDEF SYSPRINT TERMINAL
FILEDEF SYSOUT TERMINAL
FILEDEF OUTNAME DISK TRANOUT DATA (RECFM V
FILEDEF FILE1OUT DISK FILE1OUT DATA (RECFM F LRECL 20 BLKSIZE 20
FILEDEF FILE2OUT DISK FILE2OUT DATA (RECFM F LRECL 20 BLKSIZE 20
XPORT
```

## z/OS

The COBOL and C examples for z/OS:

- Compile and link edit the OUTMOD routine from the input source language statements included in the SYSIN data stream

- Place the resulting load module in a specified load library

## Generate a COBOL OUTMOD Routine

The following COBOL example is in <dbcpfx>.SAMPLIB (CHKTRANB) on the release tape.

The procedure executes the COBUCL exec that compiles the input source using program product IKFCBL00 and then link edits the object module into the USER loadlib as member Chktran.

```
//USERJOBUSER JOB (20750000), 'USERNAME', MSGCLASS=A, NOTIFY=USER,
// CLASS=B, MSGLEVEL=(1,1), REGION=5120K
//COBCOMPL EXEC COBUCL
//COB.SYSIN DD *
  IDENTIFICATION DIVISION.
  PROGRAM-ID. CHKTRAN.
  AUTHOR. USER.
  INSTALLATION. TERADATA.
  DATE-WRITTEN. 12 AUGUST 1992
  DATE COMPLIED.
  SECURITY. OPEN.
  REMARKS.
  THIS PROCEDURE IS INVOKED BY THE TERADATA FASTEXPORT
  UTILITY TO PROCESS THE RESPONSE DATA RETURNED FROM
  THE SAMPLE SELECT. THE PROCEDURE EXAMINES EACH
  RESPONSE RECORD TO DETERMINE IF THE RECORD SHOULD BE
  WRITTEN TO AN ERROR DATA SET AND THEN EITHER DROPPED
  OR WRITTEN TO THE STANDARD DATA SET. ONE ERROR DATA
  SET CONTAINS RECORDS WITH A NULL REGION CODE. THE
  OTHER ERROR DATA SET CONTAINS RECORDS WITH A TOTAL
  SALES VALUE OF LESS THAN $100. THESE LATTER RECORDS
  ARE NOT WRITTEN TO THE STANDARD DATA SET.
  ENVIRONMENT DIVISION.
  CONFIGURATION SECTION.
  SOURCE-COMPUTER. IBM-370.
  OBJECT-COMPUTER. IBM-370.
  INPUT-OUTPUT SECTION.
  FILE-CONTROL.
    SELECT SALES-DROPPED-FILE ASSIGN TO FILE1OUT.
    SELECT BAD-REGN-SALES-FILE ASSIGN TO FILE2OUT.
  DATA DIVISION.
  FILE SECTION.
  FD SALES-DROPPED-FILE
    BLOCK CONTAINS 160 RECORDS
    LABEL RECORDS STANDARD.
  01 DROPPED-TRANLOG.
    02 INDICATORS PIC 9.
    02 REGN PIC XXX.
    02 PRODUCT PIC X(8).
    02 QTY PIC S9(8) COMP.
    02 PRICE PIC S9(8) COMP.
  FD BAD-REGN-SALES-FILE
    BLOCK CONTAINS 160 RECORDS
    LABEL RECORDS STANDARD.
  01 BAD-REGN-TRANLOG.
    02 INDICATORS PIC 9.
    02 REGN PIC XXX.
```

```

02 PRODUCT      PIC X(8) .
02 QTY          PIC S9(8) COMP.
02 PRICE        PIC S9(8) COMP.
LINKAGE SECTION.
01 ENTRY-TYPE   PIC S9(5) COMP.
01 STATEMENT-NOPIC S9(5) COMP.
01 RECORD-SIZE PIC S9(5) COMP.
01 TRANLOG.
05 INDICATORS  PIC 9.
05 REGN        PIC XXX.
05 PRODUCT     PIC X(8) .
05 QTY         PIC S9(8) COMP.
05 PRICE       PIC S9(8) COMP.
01 OUTPUT-LENGTHPIC S9(5) COMP.
01 OUTPUT-AREA PIC XXXX.
PROCEDURE DIVISION USING
    ENTRY-TYPE, STATEMENT-NO, RECORD-SIZE, TRANLOG,
    OUTPUT-LENGTH, OUTPUT-AREA.
BEGIN.
MAIN.
    IF ENTRY-TYPE = 1 THEN
        OPEN OUTPUT SALES-DROPPED-FILE
        OPEN OUTPUT BAD-REGN-SALES-FILE
        GOBACK.
    IF ENTRY-TYPE = 2 THEN
        CLOSE SALES-DROPPED-FILE
        CLOSE BAD-REGN-SALES-FILE
        GOBACK.
    IF ENTRY-TYPE = 3 THEN
        PERFORM TYPE-3
        GOBACK.
    IF ENTRY-TYPE = 4 THEN
        GOBACK.
    IF ENTRY-TYPE = 5 THEN
        CLOSE SALES-DROPPED-FILE
        OPEN OUTPUT SALES-DROPPED-FILE
        CLOSE BAD-REGN-SALES-FILE
        OPEN OUTPUT BAD-REGN-SALES-FILE
        GOBACK.
    IF ENTRY-TYPE = 6 THEN
        OPEN OUTPUT SALES-DROPPED-FILE
        OPEN OUTPUT BAD-REGN-SALES-FILE
        GOBACK.
    DISPLAY "Invalid entry code = " ENTRY-TYPE.
    GOBACK.
TYPE-3.
    IF QTY IN TRANLOG * PRICE IN TRANLOG < 100 THEN
        MOVE 0 TO RECORD-SIZE
        WRITE DROPPED-TRANLOG FROM TRANLOG
    ELSE
        PERFORM TEST-NULL-REGN.
TEST-NULL-REGN.
    IF REGN IN TRANLOG = SPACES
        MOVE 999 TO REGN IN TRANLOG
        WRITE BAD-REGN-TRANLOG FROM TRANLOG.
/*
//LKED.SYSLMOD DD DSN=USERLOADLIB(CHKTRAN),DISP=MOD
//LKED.SYSIN DD *
MODE AMODE(24) RMODE(24)

```

```

        ENTRY CHKTRAN
        NAME CHKTRAN(R)
/*

```

## Generate a PL/I INMOD Routine

Following is an example of compiling and linking a PL/I INMOD routine.

```

//MXM049A JOB (22150000), 'MXM',MSGCLASS=A,NOTIFY=MXM
//STEP1 EXEC PGM=IEV90,REGION=1024K,
//          PARM='OBJ,NODECK,XREF(FULL)'
//SYSLIB DD DSN=SYS1.MACLIB,DISP=SHR
//          DD DSN=SYS1.AMODGEN,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSTEM DD SYSOUT=*
//SYSUT1 DD UNIT=VIO,SPACE=(CYL,(9,5))
//SYSUT2 DD UNIT=VIO,SPACE=(CYL,(9,5))
//SYSUT3 DD UNIT=VIO,SPACE=(CYL,(9,5))
//SYSLIN DD DSN=&&PLIA,DISP=(,PASS),UNIT=SCR,
//          SPACE=(CYL,(1,1)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSIN DD *
        TITLE 'DYNAMN'
DYNAMN CSECT
        EXTRN PLISTART
        B START-*(,R15)          BRANCH AROUND CONSTANTS
        DC AL1(L'PLIAFLAG)      LENGTH OF CONSTANTS
PLIAFLAG DC C'ASSEMBLED AT &SYSTIME ON &SYSDATE.. PLIA'
        DC C' COPYRIGHT (C) 1999 NCR CORPORATION, '
        DC C' ALL RIGHTS RESERVED.'
=====
* ENTRY POINT
=====
START SAVE (14,12)
        LR R12,R15          -> PROGRAM ENTRY POINT
        USING DYNAMN,R12
*
        LA R10,SAVAREA
        ST R10,8(R13)      FORWARD CHAIN
        ST R13,4(R10)     BACK CHAIN
        LR R13,R10
*
        LR R4,R1          SAVE PARM LIST ADDRESS
        L R3,0(,R1)       -> COMMAND WORD
        L R3,0(,R3)      COMMAND WORD
        CH R3,=H'0'      INITIAL CALL?
        BE DO_INIT      YES , DO INITIAL CODE
        CH R3,=H'6'      INITIAL CALL?
        BE DO_INIT      YES , DO INITIAL CODE
        CH R3,=H'2'      INITIAL CALL?
        BNE DO_CALL     NO, JUST GO CALL PROGRAM
=====
* SETUP PL/I ENVIRONMENT
=====
DO_INIT DS 0H
*
* WTO 'PRIOR TO INIT REQUEST'
*

```

```

MVC PRP_REQUEST,INIT INDICATE THE INIT REQUEST
*
LA R1,EXEC_ADDR GET THE PARM ADDR LIST
ST R1,EPL_EXEC_OPTS SAVE IN EPL
*
LA R1,PARM_EPL R1 --> POINTER --> REQUEST LIST
L R15,PSTART PL/I ENTRY ADDR
BALR R14,R15 INVOKE PL/I
*
WTO 'AFTER INIT REQUEST'
*=====
* CALL "OPTIONS( MAIN )" INMOD
*=====
DO_CALL DS 0H
*
WTO 'PRIOR TO CALL REQUEST'
*
MVC PRP_REQUEST,CALL INDICATE THE CALL REQUEST
*
ST R4,EPL_PROG_PARMS SAVE PARM ADDR IN EPL
LA R1,PARM_EPL R1 --> POINTER --> REQUEST LIST
L R15,PSTART PL/I ENTRY ADDR
BALR R14,R15 INVOKE PL/I
*
CH R3,=H'5' FINAL CALL?
BNE DO_RTN NO, JUST RETURN TO CALLER
*=====
* TERMINATE THE PL/I ENVIRONMENT
*=====
DO_TERM DS 0H
*
ST R15,RETCODE SAVE PL/I RETURN CODE
*
WTO 'PRIOR TO TERM REQUEST'
*
MVC PRP_REQUEST,TERM INDICATE A TERM COMMAND
*
LA R1,0 NO PARM LIST IS PRESENT
ST R1,EPL_PROG_PARMS SAVE IN EPL
*
LA R1,PARM_EPL R1 --> POINTER --> REQUEST LIST
L R15,PSTART PL/I ENTRY ADDR
BALR R14,R15 INVOKE PL/I
*
WTO 'AFTER TERM REQUEST'
*=====
* RETURN TO CALLER
*=====
DO_RTN DS 0H
*
L R13,SAVAREA+4
L R14,12(R13)
L R15,RETCODE
LM R0,R12,20(R13)
BR R14 RETURN TO YOUR CALLER
EJECT
EJECT
*=====
* CONSTANTS AND WORKAREAS

```

Appendix C: INMOD, OUTMOD and Notify Exit Routine Examples  
z/OS

```

*=====
SAVAREA  DS    20F
RETCODE  DC    F'0'
PARM_EPL DC    A(X'80000000'+IBMBZPRP)      PARAMETER ADDR LIST
PSTART   DC    A(PLISTART)
*=====
*   REQUEST STRINGS ALLOWED IN THE INTERFACE
*=====
INIT     DC    CL8'INIT'      INITIALIZE THE PROGRAM ENVIR
CALL     DC    CL8'CALL'      INVOKE THE APPL - LEAVE ENVIR UP
TERM     DC    CL8'TERM'      TERMINATE ENVIRONMENT
EXEC     DC    CL8'EXECUTE'    INIT, CALL, TERM - ALL IN ONE
*=====
*   PARAMETER LIST PASSED BY A PRE-INITIALIZED PROGRAM
*   ADDRESSED BY REG 1 = A(A(IBMBZPRP))
*   SEE IBMBZEPL DSECT.
*=====
IBMBZPRP          DS    0F
PRP_LENGTH        DC    H'16'    LEN OF THIS PRP PASSED (16)
PRP_ZERO          DC    H'0'     MUST BE ZERO
PRP_REQUEST       DC    CL8' '    'INIT' - INITIALIZE PL/I
*                  'CALL' - INVOKE APPLICATION
*                  'TERM' - TERMINATE PL/I
*                  'EXECUTE' - INIT, CALL, TERM
*
PRP_EPL_PTR       DC    A(IBMBZEPL)  A(EPL) - EXTENDED PARM LIST
*=====
*   PARAMETER LIST FOR THE PRE-INITIALIZED PROGRAM
*=====
IBMBZEPL          DS    0F
EPL_LENGTH        DC    A(EPL_SIZE)  LENGTH OF THIS EPL PASSED
EPL_TOKEN1        DC    F'0'        FIRST ENV TOKEN
EPL_TOKEN2        DC    F'0'        SECOND ENV TOKEN
EPL_PROG_PARMS    DC    F'0'        A(PARM ADDRESS LIST) ...
EPL_EXEC_OPTS     DC    A(EXEC_ADDR)  A(EXECUTION TIME OPTNS) ...
EPL_ALTMMAIN      DC    F'0'        A(ALTERNATE MAIN)
EPL_SERVICE_VEC   DC    A(IBMBZSRV)   A(SERVICE ROUTINES VECTOR)
EPL_SIZE          EQU    *-IBMBZEPL   THE SIZE OF THIS BLOCK
*=====
*   SERVICE ROUTINE VECTOR
*=====
IBMBZSRV          DS    0F
SRV_SLOTS         DC    F'2'        COUNT OF SLOTS DEFINED
SRV_USERWORD      DC    A(SRV_UA)    USER WORD
SRV_WORKAREA      DC    A(SRV_WA)    A(WORKAREA)
SRV_LOAD          DC    F'0'        A(LOAD ROUTINE)
SRV_DELETE        DC    F'0'        A(DELETE ROUTINE)
SRV_GETSTOR       DC    F'0'        A(GET STORAGE ROUTINE)
SRV_FREESTOR      DC    F'0'        A(FREE STORAGE ROUTINE)
SRV_EXCEP_RTR     DC    F'0'        A(EXCEPTION ROUTER SERVICE)
SRV_ATTN_RTR      DC    F'0'        A(ATTENTION ROUTER SERVICE)
SRV_MSG_RTR       DC    F'0'        A(MESSAGE ROUTER SERVICE)
SRV_END           DS    0F
*=====
*   SERVICE ROUTINE USERAREA
*=====
SRV_UA            DS    8F
*=====
*   SERVICE ROUTINE WORKAREA

```

```

*=====
SRV_WA          DS  0D
                DC  F'256'   LENGTH OF WORKAREA
                DS  63F     ACTUAL WORKAREA
*=====
*   EXECUTION TIME PARAMETERS
*=====
EXEC_ADDR DC    A(X'80000000'+EXEC_LEN)
EXEC_LEN  DC    AL2(EXEC_OLEN)
EXEC_OPTS DC    C'NATLANG(ENU),NOSTAE'
EXEC_OLEN EQU   *-EXEC_OPTS
*
      LTORG
R0      EQU    0
R1      EQU    1
R2      EQU    2
R3      EQU    3
R4      EQU    4
R5      EQU    5
R6      EQU    6
R7      EQU    7
R8      EQU    8
R9      EQU    9
R10     EQU    10
R11     EQU    11
R12     EQU    12
R13     EQU    13
R14     EQU    14
R15     EQU    15
      END
//*
//STEP2      EXEC IEL1CL
//PLI.SYSPRINT DD SYSOUT=*
//PLI.SYSIN   DD *
INMDPL2: PROCEDURE (X,Y) OPTIONS (MAIN);

      DCL X FIXED, Y FIXED;

      DCL 1 PARM_LIST ALIGNED BASED(P),
          10 STATUS      FIXED BINARY (31,0),
          10 RLENGTH     FIXED BINARY (31,0),
          10 BUFFER      CHAR(80);
      DCL 1 PARM_PARM2 ALIGNED BASED(Q),
          10 SEQ         FIXED BINARY (31,0),
          10 LEN         FIXED BINARY (15,0),
          10 PARAMETER   CHAR(80);
      DCL COUNT STATIC FIXED BINARY (31,0),
          INSRWS STATIC FIXED BINARY (31,0),
          REJRWS STATIC FIXED BINARY (31,0);
      DCL I, NOTMATCH FIXED BINARY (31,0);
      DCL ADDR BUILTIN, SUBSTR BUILTIN;
      DCL P POINTER, Q POINTER;
      DCL SYSPRINT FILE OUTPUT;

      P = ADDR(X);
      Q = ADDR(Y);

      OPEN FILE(SYSPRINT);
      PUT SKIP LIST('### INSIDE PL/I INMOD ROUTINE...');

```

Appendix C: INMOD, OUTMOD and Notify Exit Routine Examples  
z/OS

```

PUT SKIP LIST('STATUS =');
PUT LIST(P->STATUS);
PUT SKIP LIST('LENGTH =');
PUT LIST(P->RLENGTH);
PUT SKIP LIST('BUFFER =');
PUT LIST(SUBSTR(P->BUFFER,1,30));
PUT SKIP LIST('SEQ =');
PUT LIST(Q->SEQ);
PUT SKIP LIST('PARM =');
PUT LIST(SUBSTR(Q->PARAMETER,1,Q->LEN));

SELECT (P->STATUS);

    WHEN (6) DO; /* INITIALIZE */
        COUNT = 0;
        REJROWS = 0;
        INSROWS = 0;
        P->STATUS = 0;
    END;

    WHEN (7) DO; /* PROCESS */
        COUNT = COUNT + 1;
        NOTMATCH = 0;
        P->STATUS = 0;
        DO I = 1 TO Q -> LEN;
            IF SUBSTR(P->BUFFER,I,1) ^= SUBSTR(Q->PARAMETER,I,1)
                THEN DO;
                    NOTMATCH= 1;
                    LEAVE;
                END;
        END;
        IF NOTMATCH = 1
            THEN DO;
                PUT SKIP LIST('-----> REJECTED <-----');
                REJROWS = REJROWS + 1;
                P->RLENGTH = 0;
            END;
        ELSE
            DO;
                PUT SKIP LIST('-----> ACCEPTED <-----');
                INSROWS = INSROWS + 1;
            END;
        END;

    WHEN (5) DO; /* FINALIZE */
        P->STATUS = 0;
    END;

    OTHERWISE DO;
        PUT SKIP LIST ('UNKNOWN CODE...');
        P->STATUS = 99;
    END;

END;

PUT SKIP LIST('STATUS =');
PUT LIST(P->STATUS);
PUT SKIP LIST('LENGTH =');
PUT LIST(P->RLENGTH);
PUT SKIP LIST('TOTAL =');

```



```

PUT LIST (COUNT);
PUT SKIP LIST ('INSERTS =');
PUT LIST (INSROWS);
PUT SKIP LIST ('REJROWS =');
PUT LIST (REJROWS);
PUT SKIP LIST ('-----');
CLOSE FILE (SYSPRINT);

END INMDPL2;
//LKED.SYSPRINT DD SYSOUT=*
//LKED.PLIA DD DISP=(OLD,DELETE),DSN=&&PLIA
//LKED.SYSIN DD *
INCLUDE PLIA
ENTRY DYNAMN
NAME INMDPL2(R)

```

## Generate a SAS/C OUTMOD Routine

The following SAS/C example is in <dbcpfx>.SAMPLIB (CHKTRANC) on the release tape.

The procedure executes the SAS/C exec LC375CL, which compiles the source statements and then link edits them into the USER loadlib as member Chktran.

The ENTRY=NONE parameter to the LC375CL proc is required because the entry point name of the OUTMOD routine is not \_dynamn. The ENTRY parameter to the linkedit is also required because the LC375CL proc does not generate such a parameter.

```

//USEREXP JOB (20750000),'USERNAME',MSGCLASS=A,NOTIFY=USER,
// CLASS=B,MSGLEVEL=(1,1),REGION=5120K
//CCOMPL EXEC LC375CL,ENTRY=NONE
//C.SYSIN DD DATA,DLM=##
/* TITLE CHKTRAN ... Output procedure for TranLogTable export run */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*
/* Purpose This procedure is called by the Teradata FastExport */
/* utility for each response row returned to the host. */
/* The procedure examines each row to determine if the */
/* row should be output to an error data set and then */
/* either dropped or written to the standard data set. */
/* One error data sets contain records with a null */
/* region code the other contains records with a total */
/* sales value less than $100. These latter records */
/* are not written to the standard data set. */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#include <stdio.h>
#define chkbit(a,b) (((0x80 >> (b-1)) & (a)) ? 1 : 0)
/* Define the structure of the response row */
struct tranlog
{
char Indicators;
char Region[3];
char Product[8];
long Qty;
long Price;

```

```

    };#define minsale      100    /* minimum sales to report */
FILE *file1, *file2;
int  chktran(EntryType, StmtNo,
             RespLen, RespRec,
             OutLen, OutRec)
int    *EntryType;
int    *StmtNo;
int    *RespLen;
struct tranlog*RespRec;
int    *OutLen;
char   *OutRec;
{
long recsize;
struct tranlog recout;
/* case on entry type */
switch (*EntryType) {
case 1:
    /* Normal start */
    file1 = fopen("ddn:file1out", "wb");
    file2 = fopen("ddn:file2out", "wb");
    break;
case 2:
    /* EOF for response data */
    fclose(file1);
    fclose(file2);
    break;
case 3:
    /* Process response record */
    if ((RespRec->Qty * RespRec->Price) < minsale)
    {
        *RespLen = 0;
        recout = *RespRec;
        recsize = fwrite(&recout, sizeof(recout), 1, file1);
    }
    else
    {
        /* If the region is null then change it to a dummy region.
        */ if (chkbit(RespRec->Indicators,1))
        {
            RespRec->Region[1] = '9';
            RespRec->Region[2] = '9';
            RespRec->Region[3] = '9';
            recout = *RespRec;
            recsize = fwrite(&recout, sizeof(recout), 1, file2);
        }
    }
    break;
case 4:
    /* no checkpoints to worry about */
    break;
case 5:
    /* DBC restart - close and reopen the output files */
    file1 = freopen("ddn:file1out", "wb", file1);
    file2 = freopen("ddn:file2out", "wb", file2);
    break;
case 6:
    /* Host restart same as normal since there are no checkpoints */
    file1 = fopen("ddn:file1out", "wb");
    file2 = fopen("ddn:file2out", "wb");

```

```

        break;
default:
    printf("Invalid entry code = %d\n", *EntryType);
    break;
}
return(0);
}
##
//LKED.SYSLMOD DD DSN=USER.LIB.LOAD(CHKTRAN),DISP=MOD
//LKED.SYSIN DD DATA,DLM=##
MODE AMODE(24) RMODE(24)
ENTRY CHKTRAN
NAME CHKTRAN(R)
##

```

## Execute the OUTMOD Routine

Use the following z/OS JCL to execute FastExport and either the SAS/C or the COBOL OUTMOD routine. The JOBLIB statements define both run-time libraries.

The SAS/C procedures are required to support FastExport. The COBOL procedures are required only if an OUTMOD routine written in COBOL exists.

The FastExport control commands in this example vary from those for the z/VM example as follows:

- The LOGON statement is included with the rest of the FastExport control commands
- The IMPORT command is omitted, meaning that the Teradata SQL SELECT statement was modified to include a date range in the request rather than acquiring it through a USING clause

```

//USEREXP JOB (20750000), 'USERNAME',MSGCLASS=A,NOTIFY=USER,
//          CLASS=A,MSGLEVEL=(1,1),REGION=4096K
//JOBLIB   DD DSN=TERADATA.TRLOAD,DSP=SHR
//          DD DSN=TERADATA.APPLOAD,DISP=SHR
//          DD DSN=SYS1.VSCOLIB,DISP=SHR
//EXP      EXEC PGM=XPORT,REGION=4096K
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSTEM   DD SYSOUT=*
//OUTNAME  DD DSN=USER.EXPORT.TEST1,DISP=(NEW,CATLG,DELETE),
//          DCB=(NCP=20,RECFM=FB,LRECL=20,BLKSIZE=3200,DSORG=PS),
//          UNIT=SYSDA,SPACE=(3200,(300,1))
//FILE1OUT DD DSN=USER.EXPORT.TEST2,DISP=(NEW,CATLG,DELETE),
//          DCB=(NCP=20,RECFM=FB,LRECL=20,BLKSIZE=3200,DSORG=PS),
//          UNIT=SYSDA,SPACE=(3200,(1,1))
//FILE2OUT DD DSN=USER.EXPORT.TEST3,DISP=(NEW,CATLG,DELETE),
//          DCB=(NCP=20,RECFM=FB,LRECL=20,BLKSIZE=3200,DSORG=PS),
//          UNIT=SYSDA,SPACE=(3200,(1,1))
//SYSIN    DD *
.LOGTABLE TranLog;          /* define restart log          */
.LOGON TDPV/USER,USER;     /* the logon string            */
.BEGIN EXPORT              /* Specify export function     */
SESSIONS 20;              /* number of sessions to be used */
.EXPORT OUTFILE OutName   /* identify the destination    */
OUTMOD ChkTran;          /* file and the procedure      */
                          /* to receive the records.     */

```

```
SELECTRegion,          /* provide the SQL SELECT      */
   ProdCode,
   Quantity,
   Price
From TranLogTable
WHERE TranDate BETWEEN 911231 and 920701
ORDER BY Region, ProdCode;
.END EXPORT;          /* terminate FastExport      */
.LOGOFF              /* disconnect from the DBC    */
```

## UNIX

### C INMOD Example

This INMOD example reads the function code and executes different processing functions based on its value.

```
/* This program is for release 4.1 MULTILOAD INMOD testing using C
   user exit routine.
   When this routine is activated it looks at the content of the
   function code passed (a->code) and depending on its value, it
   0) initializes, i.e., opens a file, etc...
   1) reads a record
   5) acknowledges "close inmod" request. The user exit routine
   must return "return code"(a->code) and "length" (a->len). You
   should send return code = zero when no errors occur and non-zero for
   an error. MULTILOAD expects length = zero at the end of file. Then
   it sends "CLOSE INMOD" request. THE USER EXIT routine must
   explicitly return "return code" = ZERO to terminate the
   conversation. */
#include <stddef.h>
#include <stdlib.h>
#include <stdio.h>
typedef unsigned short   Int16;
typedef unsigned char    Int8;
typedef unsigned long int Int32;

/* PASSING parameter structures */

typedef struct {
    Int32 code;
    Int32 len;
    Int8  buf[80];
} inmodbuf;

typedef struct {
    Int32 seq;
    Int16 len;
    char  param[80];
} inmodpty;

static FILE *IN;
static int count=0;
char *memcpy();
```

```

void _dynamn(a,b)
inmodbuf *a;
inmodpty *b;
{int code=0;
char tempbuf[80];

memcpy(tempbuf,a->buf,sizeof(a->buf));
tempbuf[79]='\0';
printf("BEGIN--> %d %d %s\n",a->code,a->len,tempbuf);
printf("      +++ %d %d %s\n",b->seq ,b->len,b->param);
code= (int) a->code;
switch (code) {
case 0:
/* Here you open the file and read the first record */
printf("## CODE=0, opening...\n");
IN=fopen("ddn:INDATA","rb");
if (! ferror(IN)) {
if (! readrecord(a))
fclose(IN);
};
break;
case 1:
/* MultiLoad requested next record, read it */
printf("## CODE=1, reading...\n");
if (! readrecord(a))
fclose(IN);
break;
case 5:
/* MultiLoad is closing INMOD routine */
a->code=0;
a->len=0;
printf("## CODE=5, terminating...\n");
break;
default:
a->code=12; /* any number not = to zero */
a->len=0;
printf("##### UNKNOWN code #####\n");a->code=0;a->len=0;
};

memcpy(tempbuf,a->buf,sizeof(a->buf));
tempbuf[79]='\0';
printf("END --> %d %d %s\n",a->code,a->len,tempbuf);
printf("      +++ %d %d %s\n",b->seq ,b->len,b->param);
}

vint readrecord(a)
inmodbuf *a;
{
int rtn=0;
char tempbuf[80];

if (fread((char *)&(a->buf),sizeof(a->buf),1,IN)) {
count++;

memcpy(tempbuf,a->buf,sizeof(a->buf));
tempbuf[79]='\0';
printf("      %d %s \n",count,tempbuf);
}
}

```

```

    a->len=80;
    a->code=0;
    rtn=1;
};
if ferror(IN) {
    printf("==== error ====\n");
    a->code=16; /* any non zero number */
    a->len=0;
};
if feof(IN) { /* EOF, set length = zero */
    printf("==== EOF ====\n");
    a->code=9;
    a->len=9;
};
return(rtn);
}

```

## C Notify Exit Parameters

Following is a C structure that describes the parameters passed to a notify exit routine. The pointer is a 32-bit value, and all long definitions are 32-bit unsigned.

```

typedef unsigned long UInt32;
typedef enum {
    NMEventInitialize      = 0,
    NMEventFileInmodOpen  = 1,
    NMEventPhaseIBegin    = 2,
    NMEventCheckPoint     = 3,
    NMEventPhaseIEnd      = 4,
    NMEventPhaseIIBegin   = 5,
    NMEventPhaseIIEnd     = 6,
    NMEventErrorTableI    = 7,
    NMEventErrorTableII   = 8,
    NMEventDBSRestart     = 9,
    NMEventCLIErrror      = 10,
    NMEventDBSErrror      = 11,
    NMEventExit            = 12,
    NMEventAmpsDown       = 21,
    NMEventImportBegin    = 22,
    NMEventImportEnd      = 23,
    NMEventDeleteInit     = 24,
    NMEventDeleteBegin    = 25,
    NMEventDeleteEnd      = 26,
    NMEventDeleteExit     = 27
} NfyMLDEvent;
/*****
/* Structure for User Exit Interface */
/* DR42570 - redesigned and rewritten */
*****/
#define NOTIFYID_FASTLOAD      1
#define NOTIFYID_MULTILOAD    2
#define NOTIFYID_FASTEXPORT    3
#define NOTIFYID_BTEQ         4
#define NOTIFYID_TPUMP        5
#define MAXVERSIONIDLEN      32
#define MAXUTILITYNAMELEN    32
#define MAXUSERNAMELEN      64
#define MAXUSERSTRLEN        80

```

```

#define MAXTABLENAMELEN 128
#define MAXFILENAMELEN 256
typedef struct _MLNotifyExitParm {
    UInt32 Event; /* should be NfyMLDEvent values */
    union {
        struct {
            UInt32 VersionLen;
            char VersionId[MAXVERSIONIDLEN];
            UInt32 UtilityId;
            UInt32 UtilityNameLen;
            char UtilityName[MAXUTILITYNAMELEN];
            UInt32 UserNameLen;
            char UserName[MAXUSERNAMELEN];
            UInt32 UserStringLen;
            char UserString[MAXUSERSTRLEN];
        } Initialize;
        struct {
            UInt32 FileNameLen;
            char FileOrInmodName[MAXFILENAMELEN];
            UInt32 ImportNo;
        } FileInmodOpen ;
        struct {
            UInt32 TableNameLen;
            char TableName[MAXTABLENAMELEN];
            UInt32 TableNo;
        } PhaseIBegin;
        struct {
            UInt32 RecordCount;
        } CheckPoint;
        struct {
            UInt32 RecsRead;
            UInt32 RecsSkipped;
            UInt32 RecsRejected;
            UInt32 RecsSent;
        } PhaseIEnd ;
        struct {
            UInt32 dummy;
        } PhaseIIBegin;
        struct {
            UInt32 Inserts;
            UInt32 Updates;
            UInt32 Deletes;
            UInt32 TableNo;
        } PhaseIIEnd;
        struct {
            UInt32 Rows;
            UInt32 TableNo;
        } ErrorTableI;
        struct {
            UInt32 Rows;
            UInt32 TableNo;
        } ErrorTableII ;
        struct {
            UInt32 dummy;
        } DBSRestart;
        struct {
            UInt32 ErrorCode;
        } CLIErrors;
        struct {

```

```
        UInt32 ErrorCode;
    } DBSError;
    struct {
        UInt32 ReturnCode;
    } Exit;
    struct {
        UInt32 dummy;
    } AmpsDown;
    struct {
        UInt32 ImportNo;
    } ImportBegin ;
    struct {
        UInt32 RecsRead;
        UInt32 RecsSkipped;
        UInt32 RecsRejected;
        UInt32 RecsSent;
        UInt32 ImportNo;
    } ImportEnd ;
    struct {
        UInt32 dummy;
    } DeleteInit;
    struct {
        UInt32 TableNameLen;
        char    TableName[MAXTABLENAMELEN];
        UInt32 TableNo;
    } DeleteBegin;
    struct {
        UInt32 Deletes;
        UInt32 TableNo;
    } DeleteEnd;
    struct {
        UInt32 ReturnCode;
    } DeleteExit;
    } Vals;
} MLNotifyExitParm;

#ifdef I370
#define MLNfyExit MLNfEx
#endif
extern long MLNfyExit(
#ifdef __STDC__
        MLNotifyExitParm *Parms
#endif
);
```

## Compile and Link Routines

**Note:** For a description of the syntax diagrams used in this book, see [Appendix A: “How to Read Syntax Diagrams.”](#)

### Sun Solaris Opteron

To compile and link source files into a shared object module for INMOD or notify exit routines on Sun Solaris Opteron client systems, use the following syntax:



```
cc -dy -G sourcefile.c -o shared-object-name
```

2409A055

where

Syntax Element	Description
cc	Invokes the MetaWare High C Compiler
-dy	Specifies to use dynamic linking
-G	Specifies to create a shared object
<i>sourcefile</i>	Is a source module for the INMOD
-o	Specifies the output file name
<i>shared-object-name</i>	Specifies the resulting shared object module This is the name specified as the: <ul style="list-style-type: none"> <li>INMOD <i>modulename</i> parameter in the <b>IMPORT</b> of the FastExport job script</li> <li>EXIT <i>name</i> parameter of the NOTIFY option in the <b>BEGIN EXPORT</b> command of the FastExport job script</li> </ul> The <i>shared-object-name</i> can be any valid UNIX file name.

## MP-RAS and Sun Solaris SPARC

To compile and link source files into a shared object module for INMOD or notify exit routines on MP-RAS and Sun Solaris SPARC client systems, use the following syntax:

### Compile Syntax

```
gcc -shared -fPIC sourcefile.c -o shared-object-name
```

2410B001

where

Syntax Element	Description
gcc	Call to the program that invokes the native UNIX C compiler
-shared	Flag that produces a shared object that can then be linked with other objects to form an executable
-fPIC	Compiler option that generates Position Independent Code for all user exit routines

Syntax Element	Description
<i>sourcefile</i>	UNIX file name of the source file for the INMOD or notify exit routine
-o	Switch to the linker
<i>shared-object-name</i>	Name of the shared object file This is the name specified as the: <ul style="list-style-type: none"> <li>INMOD <i>modulename</i> parameter in the <a href="#">IMPORT</a> of the FastExport job script</li> <li>EXIT <i>name</i> parameter of the NOTIFY option in the <a href="#">BEGIN EXPORT</a> command of the FastExport job script</li> </ul> The <i>shared-object-name</i> can be any valid UNIX file name.

### HP-UX PA RISC

To compile and link source files into a shared object module for INMOD and notify exit routines on HP-UX PA RISC client systems, use the following syntax:

#### Compile Syntax

```
cc +z — +ul — -c — sourcefile.c —▶
```

2409A006

#### Link Syntax

```
ld -b — objectfile.o — -o shared-object-name —▶
```

2409A007

where

Syntax Element	Description
-b	Linker option that generates a shared object file
-c	Compile-only option (does not link)
cc	Call to the program that invokes the native UNIX C compiler
ld	Call to the program that invokes the native UNIX linker
-o	Switch to the linker
<i>objectfile</i>	Compiler-generated file used by the linker to generate <i>shared-object-name</i>

Syntax Element	Description
<i>shared-object-name</i>	Name of the shared object file This is the name specified as: <ul style="list-style-type: none"> <li>The INMOD <i>modulename</i> parameter of the <b>IMPORT</b> command of the FastExport job script</li> <li>The EXIT <i>name</i> parameter for the NOTIFY option of the <b>BEGIN EXPORT</b> command of the FastExport job script</li> </ul> The <i>shared-object-name</i> can be any valid UNIX file name.
<i>sourcefile</i>	UNIX file name(s) of the source file(s) for the INMOD or notify exit routine
+ul	Compiler option that allows pointers to access non-natively aligned data
+z	Compiler option that generates Position Independent Code for all user exit routines

### HP-UX Itanium

Use the following syntax example to compile a C INMOD on HP-UX Itanium-based clients.

#### Compile Syntax

```
cc — +u1 — -D_REENTRANT — +DD64 — -c — inmod.c —▶
```

2409A057

where

Syntax Element	Definition
cc	Invokes the MetaWare High C compiler
+u1	Is a compiler option that allows pointers to access non-natively aligned data
-D_REENTRANT	Ensures that all the Pthread definitions are visible at compile time
+DD64	Generates 64-bit object code for PA2.0 architecture
-c	Compiles one or more source files but does not enter the linking phase
<i>inmod.c</i>	A C source module for the INMOD

Use the following syntax example to link the object modules on HP-UX Itanium into the shared object.

### Link Syntax

ld — -n — -b — *inmod.o* — -lc — -o — *inmod.so* —▶

2409A056

where

Syntax Element	Definition
ld	Invokes the UNIX linker editor
-n	Generates an executable with file type SHARE_MAGIC. This option is ignored in 64-bit mode.
-b	Is a linker option specified to generate a shared object file
<i>inmod.o</i>	Is an object module derived from the compile step (see above)
-lc	Search a library <i>libc.a</i> , <i>libc.so</i> , or <i>libc.sh</i>
-o	Specifies the output filename; default is <i>a.out</i>
<i>inmod.so</i>	Specifies the resulting shared object module This is the user-specified name in the IMPORT command.

### IBM AIX

To compile and link source files into a shared object module for INMOD and notify exit routines on IBM AIX client systems, use the following syntax:

#### Compile Syntax

cc — -c — -brtl — -fPIC — *sourcefile.c* —▶

2409B008

#### Link Syntax

ld — -G — -e\_dynamn — -bE: *export\_dynamn.txt* — (A)

(A) — *objectfile.o* — -o *shared-object-name* — -lm — -lc —▶

2409A009

where

Syntax Element	Description
-c	Compiler option specifying to not send object files to the linkage editor
cc	Call to the program that invokes the native UNIX C compiler
-bE: <i>export_dynamn.txt</i>	The linker option that exports the symbol "_dynamn" explicitly and the file <i>export_dynamn.txt</i> contains the symbol
-brtl	Tells the linkage editor to accept both <i>.sl</i> and <i>.a</i> library file types
-e_dynamn	Sets the entry point of the exit routine to <i>_dynamn</i>
-fPIC	Compiler option that generates Position Independent Code for all user exit routines
-G	Produces a shared object-enabled for use with the run-time linker
-lc	Link with the <i>/lib/libc.a</i> library
ld	Call to the program that invokes the native UNIX linker
-lm	Link with the <i>/lib/libm.a</i> library
-o	Switch to the linker
<i>objectfile</i>	Compiler-generated file used by the linker to generate <i>shared-object-name</i>
<i>shared-object-name</i>	Name of the shared object file The <i>shared-object-name</i> can be any valid UNIX file name. This is the name specified as: <ul style="list-style-type: none"> <li>The INMOD <i>modulename</i> parameter of the <b>IMPORT</b> command of the FastExport job script</li> <li>The EXIT <i>name</i> parameter for the NOTIFY option of the <b>BEGIN EXPORT</b> command of the FastExport job script</li> </ul>
<i>sourcefile</i>	UNIX file name(s) of the source file(s) for the INMOD or notify exit routine

## Linux

To compile and link source files into a shared object module for INMOD or notify exit routines on Linux client systems, use the following syntax.

**Note:** Be sure to compile the INMOD and notify exit routines in 32-bit mode so they are compatible with Teradata FastExport.

### Compile Syntax

```
gcc -shared -fPIC inmod.c inmod.so
```

2410A015

where

Syntax Element	Description
gcc	Call to the program that invokes the native C compiler
-shared	Flag that produces a shared object that can then be linked with other objects to form an executable
-fPIC	Compiler option that generates position-independent code for all user exit routines
-o	Output file name
<i>inmod.c</i>	An INMOD source file name
<i>inmod.so</i>	An INMOD shared object name

## Windows

To generate and use an INMOD, OUTMOD or notify exit routine on a Windows client system, the routine must:

- Be written in C
- Have a dynamn entry point that is a `__declspec`
- Be saved as a Dynamic Link Library (DLL) file

## Generating Routines

Three sample program files are provided with the FastExport utility software to help generate and use INMOD, OUTMOD and notify exit routines in the FastExport job scripts on network-attached Windows client systems. The listings of these sample files are presented later in this appendix:

Sample File	Description
<i>feimod.c</i>	Source file for an INMOD routine
<i>feomod.c</i>	Source file for an OUTMOD routine
<i>fenotf.c</i>	Source file for a notify exit routine

**Note:** Though they are presented as Windows examples, the three sample files are valid for UNIX systems, when compiled and linked into a shared object file as indicated in the comment banner of each file.

## To generate and use an INMOD, OUTMOD, or notify exit routine in a FastExport job

Refer to the referenced sample file listings and use this procedure for generating and using an INMOD, OUTMOD, or notify exit routine.

- 1 Edit the routine source file and ensure that the dynamn name is a `_declspec`
- 2 See the listing of the sample routine files later in this appendix:

- `feimod.c`
- `feomod.c`
- `fenotf.c`

- 3 Use the following command to create a DLL:

```
c1 /DWIN32 /LD sourcefilename
```

where *sourcefilename* is the name of the INMOD, OUTMOD, or notify exit routine source file.

Successful command execution produces a file with the same name as the source file with the `.dll` extension:

*sourcefilename.dll*

- 4 Use the *sourcefilename.dll* in the FastExport job script as follows:

Routine Type	Use the <i>sourcefilename.dll</i> File as
INMOD	INMOD <i>modulename</i> in the IMPORT command
OUTMOD	OUTMOD <i>modulename</i> in the EXPORT command
Notify Exit	EXIT <i>name</i> specification of the NOTIFY option in the BEGIN EXPORT command

## Sample INMOD Routine

Following is the listing of the *feimod.c* sample INMOD routine that is provided with the FastExport utility software:

```
#include <stdio.h>
#include <stdlib.h>

/*****
/*
/* feimod.c - Sample Inmod for FastExport.
/*
/* Purpose - This inmod generates two integers per record. The
/* first is even and the second number is odd.
/*
/* Note - The number of records per file is determined by the
/* variable NUM_RECORDS
/*
/* Execute - Build Inmod on a Unix system
/* compile and link into shared object
/* cc -G feimod.c - o feimod.so
/*
/*
```

Appendix C: INMOD, OUTMOD and Notify Exit Routine Examples  
Windows

```

/*          - Build Inmod on a Win32 system                                */
/*          compile and link into dynamic link library                    */
/*          cl /DWIN32 /LD feimod.c                                       */
/*          */                                                             */
/*****/

static int msg_cnt = 0;

typedef struct {
long   ioseq;
short  len;
char   param[2000];
} param_type;

/* This structure is used to pass an odd and an even integer */
/* back to multiload */

typedef struct {
long   code;
long   len;
char   data[32768];
} data_type;

#ifdef WIN32
/* Change for WIN32 */
_declspec(dllexport) void _dynamn(data_type *data_buf , param_type *parm_buf)
#else
void _dynamn(data_buf, parm_buf)
data_type *data_buf;
param_type *parm_buf;
#endif
{
char *myptr;
int i;

static long RECNUM = 0; /* number of records to load */
static int odd_counter = 0; /* odd integer counter */
static int even_counter = 0; /* even integer counter */

#ifdef DEBUG
printf("\n");
printf("jmod2: on input:\n");
printf("jmod2: message code: %d\n", data_buf->code);
printf("jmod2: data bytes: %d\n", data_buf->len);
if (data_buf->len)
printf("jmod2: data: *%s*\n", data_buf->data);
printf("jmod2: param ioseq: %d\n", parm_buf->ioseq);
printf("jmod2: param bytes: %d\n", parm_buf->len);
if (parm_buf->len)
printf("jmod2: param str: %s\n", parm_buf->param);
printf("jmod2: message cnt: %d\n", ++msg_cnt);
#endif

switch (data_buf->code)
{
case 0: printf("jmod2: initializing and returning 1st record:\n");
RECNUM = 6;

printf("jmod2: Records requested = %ld\n", RECNUM);

```



```

if ( RECNUM <= 0 ) {
    printf("jmod2: numbers of records is <= 0 [%ld]\n", RECNUM);
}

/* initialize the counters */
odd_counter = 1;
even_counter = 2;

/* copy the counters to the data buffer */
myptr = (char *) &odd_counter;
for (i=0; i<4; ++i, ++myptr) {
    data_buf->data[i] = *myptr;
}

myptr = (char *) &even_counter;
for (i=4; i<8; ++i, ++myptr) {
    data_buf->data[i] = *myptr;
}

/* go to next values and increment the counters */
odd_counter += 2;
even_counter +=2;
--RECNUM;
/* return the results */
data_buf->code = 0;
data_buf->len = 8;
break;

case 1:
#ifdef DEBUG
    printf("jmod2: returning a record:\n");
#endif
    if (RECNUM) {

        /* copy the counters to the data buffer */
        myptr = (char *) &odd_counter;
        for (i=0; i<4; ++i, ++myptr) {
            data_buf->data[i] = *myptr;
        }

        myptr = (char *) &even_counter;
        for (i=4; i<8; ++i, ++myptr) {
            data_buf->data[i] = *myptr;
        }

        /* increment to next values and decrement record counter */
        odd_counter += 2;
        even_counter += 2;
        --RECNUM;

        /* return the results */
        data_buf->code = 0;
        data_buf->len = 8;
        break;
    } else {

        /* done sending records, return non-zero result */
        printf("jmod2: all records sent\n");
    }
}

```

```

        data_buf->code = 1;
        break;
    }

    case 2: printf("jmod2: repositioning to last checkpoint (HOST)\n");
        data_buf->code = 0;
        data_buf->len = 0;
        break;
    case 3: printf("jmod2: taking a checkpoint\n");
        data_buf->code = 0;
        data_buf->len = 0;
        break;
    case 4: printf("jmod2: repositioning to last checkpoint (DBC)\n");
        data_buf->code = 0;
        data_buf->len = 0;
        break;
    case 5: printf("jmod2: terminating:\n");
        data_buf->code = 0;
        data_buf->len = 0;
        break;
    case 6: printf("jmod2: initializing and receiving 1st record:\n");
        data_buf->data[1] = '%';
        data_buf->code = 0;
        break;
    case 7: printf("jmod2: receiving a record:\n");
        data_buf->data[1] = '%';
        data_buf->code = 0;
        break;
    }

#ifdef DEBUG
printf("jmod2: on output:\n");
printf("jmod2: message code: %d\n", data_buf->code);
printf("jmod2: data bytes: %d\n", data_buf->len);
if (data_buf->len)
    printf("jmod2: data: %s*\n", data_buf->data);
printf("\n");
#endif
}

```

## Sample OUTMOD Routine

Following is the listing of the *feomod.c* sample OUTMOD routine that is provided with the FastExport utility software:

```

#include <stdio.h>
#include <stdlib.h>

/*****
/*
/* feomod.c - Sample Outmod for FastExport.
/*
/* Purpose - This outmod deletes the data passed to it.
/*
/* Execute - Build Outmod on a Unix system
/* compile and link into shared object
*/

```

```

/*          cc -G feomod.c - o feomod.so          */
/*          */
/*          - Build Outmod on a Win32 system      */
/*          compile and link into dynamic link library */
/*          cl /DWIN32 /LD feomod.c             */
/*          */
/*****/

typedef unsigned short      Int16;
typedef unsigned char       Int8;
typedef unsigned long int   Int32;
#ifdef WIN32
__declspec(dllexport) Int32 _dynamn( int      *code,
                                     int      *stmno,
                                     int      *InLen,
                                     char     *InBuf,
                                     int      *OutLen,
                                     char     *OutBuf
                                     )
#else
Int32 _dynamn(code, stmno, InLen, InBuf, OutLen, OutBuf)
int      *code;
int      *stmno;
int      *InLen;
char     *InBuf;
int      *OutLen;
char     *OutBuf;
#endif

{
/* case on entry code
*/
switch (*code) {

case 1: /* Initialization, no other values */
printf ("OUTMOD Initial Entry\n");
break;
case 2: /* Cleanup call, no other values */
printf ("OUTMOD End of Responses Entry\n");
break;
case 3: /* Process response record */
*InLen=0;
*OutLen=0;
break;
case 4:
/* Checkpoint, no other values */
printf ("OUTMOD Checkpoint Entry\n");
break;
case 5:
/* DBC restart - close and reopen the output files */
printf ("OUTMOD DBC Restart Entry\n");
break;
case 6:
/* Host restart */
printf ("OUTMOD Host Restart Entry\n");
break;
default:
printf ("OUTMOD Invalid Entry code\n");
}
}

```

```

        break;

    }
    return (0);
}

```

### Sample Notify Exit Routine

Following is the listing of the *fenotf.c* sample notify exit routine that is provided with the FastExport utility software:

```

/*****
/*
/* fenotf.c - Sample Notify Exit for FastExport.
/*
/* Purpose - This is a sample notify exit for FastExport.
/*
/*
/* Execute - Build Notify on a Unix system
/*             compile and link into shared object
/*             cc -G fenotf.c - o fenotf.so
/*
/*
/* - Build Notify on a Win32 system
/*             compile and link into dynamic link library
/*             cl /DWIN32 /LD fenotf.c
/* History : Updated with new events.
/*
*****/
#include <stdio.h>
typedef unsigned long UInt32;
#define NOTIFYID_FASTLOAD      1
#define NOTIFYID_MULTILOAD    2
#define NOTIFYID_FASTEEXPORT  3
#define NOTIFYID_BTEQ         4
#define NOTIFYID_TPUMP        5

#define MAXVERSIONIDLEN      32
#define MAXUTILITYNAMELEN   32
#define MAXUSERNAMELEN      64
#define MAXUSERSTRLEN       80
#define MAXFILENAMELEN      256
#define MAXREQUESTLEN       32000

typedef enum {
    NXEventInitialize      = 0,
    NXEventFileInmodOpen  = 1,
    NXEventDBSRestart     = 9,
    NXEventCLIError       = 10,
    NXEventDBSError       = 11,
    NXEventExit           = 12,
    NXEventExportBegin    = 31,
    NXEventReqSubmitBegin = 32,
    NXEventReqSubmitEnd   = 33,
    NXEventReqFetchBegin  = 34,
    NXEventFileOutmodOpen = 35,
    NXEventStmtFetchBegin = 36,
    NXEventStmtFetchEnd   = 37,
    NXEventReqFetchEnd    = 38,
    NXEventExportEnd      = 39
} NfyExpEvent;

```

```

typedef struct _FXNotifyExitParm {
    UInt32 Event;
    union {
        struct {
            UInt32 VersionLen;
            char   VersionId[MAXVERSIONIDLEN];
            UInt32 UtilityId;
            UInt32 UtilityNameLen;
            char   UtilityName[MAXUTILITYNAMELEN];
            UInt32 UserNameLen;
            char   UserName[MAXUSERNAMELEN];
            UInt32 UserStringLen;
            char   UserString[MAXUSERSTRLEN];
        } Initialize;
        struct {
            UInt32 FileNameLen;
            char   FileOrInmodName [MAXFILENAMELEN];
            UInt32 dummy;
        } FileInmodOpen ;
        struct {
            UInt32 dummy;
        } DBSRestart;
        struct {
            UInt32 ErrorCode;
        } CLIErrror;
        struct {
            UInt32 ErrorCode;
        } DBSError;
        struct {
            UInt32 ReturnCode;
        } Exit;
        struct {
            UInt32 dummy;
        } ExportBegin;
        struct {
            UInt32 RequestLen;
            char   Request[MAXREQUESTLEN];
        } ReqSubmitBegin;
        struct {
            UInt32 StatementCnt;
            UInt32 BlockCnt;
        } ReqSubmitEnd;
        struct {
            UInt32 dummy;
        } ReqFetchBegin;
        struct {
            UInt32 FileNameLen;
            char   FileOrOutmodName [MAXFILENAMELEN];
        } FileOutmodOpen;
        struct {
            UInt32 StatementNo;
            UInt32 BlockCnt;
        } StmtFetchBegin;
        struct {
            UInt32 Records;
        } StmtFetchEnd;
        struct {
            UInt32 RecsExported;
        }
    }
};

```

## Appendix C: INMOD, OUTMOD and Notify Exit Routine Examples Windows

```
        UInt32 RecsRejected;
    } ReqFetchEnd;
    struct {
        UInt32 RecsExported;
        UInt32 RecsRejected;
    } ExportEnd;
    } Vals;
} FXNotifyExitParm;

extern long FXNotifyExit(
#ifdef __STDC__
        FXNotifyExitParm *Parms
#endif
);

#ifdef WIN32
__declspec(dllexport) long __dynamn(FXNotifyExitParm *P)
#else
long __dynamn( FXNotifyExitParm *P)
#endif
{
    FILE *fp;

    if (!(fp = fopen("NFYEXIT.OUT", "a")))
        return(1);

    switch(P->Event) {
    case NXEventInitialize :    /* Nothing */
        fprintf(fp, "exit called @ fexp init.\n");
        fprintf(fp, "Version: %s\n", P->Vals.Initialize.VersionId);
        fprintf(fp, "Utility: %s\n", P->Vals.Initialize.UtilityName);
        fprintf(fp, "User: %s\n", P->Vals.Initialize.UserName);
        if (P->Vals.Initialize.UserStringLength)
            fprintf(fp, "UserString: %s\n", P->Vals.Initialize.UserString);
        break;

    case NXEventFileInmodOpen:
        fprintf(fp, "exit called @ input file open: %s\n",
            P->Vals.FileInmodOpen.FileOrInmodName);
        break;

    case NXEventDBSRestart :
        fprintf(fp, "exit called @ RDBMS restart detected\n");
        break;

    case NXEventCLIError :
        fprintf(fp, "exit called @ CLI error %d\n",
            P->Vals.CLIError.ErrorCode);
        break;

    case NXEventDBSError :
        fprintf(fp, "exit called @ DBS error %d\n",
            P->Vals.DBSError.ErrorCode);
        break;

    case NXEventExit :
        fprintf(fp,
            "exit called @ fexp notify out of scope: return code %d.\n",
            P->Vals.Exit.ReturnCode);
    }
```

```

        break;

    case NXEventExportBegin :
        fprintf(fp, "exit called @ export beginning.\n");
        break;

    case NXEventReqSubmitBegin :
        fprintf(fp, "exit called @ request submitted: '%s'.\n",
            P->Vals.ReqSubmitBegin.Request);
        break;

    case NXEventReqSubmitEnd :
        fprintf(fp, "exit called @ request done: %d statement(s), \
%d blocks.\n",
            P->Vals.ReqSubmitEnd.StatementCnt,
            P->Vals.ReqSubmitEnd.BlockCnt);
        break;

    case NXEventReqFetchBegin :
        fprintf(fp, "exit called @ request fetch beginning.\n");
        break;

    case NXEventFileOutmodOpen:
        fprintf(fp, "exit called @ output file open: %s\n",
            P->Vals.FileOutmodOpen.FileOrOutmodName);
        break;

    case NXEventStmtFetchBegin :
        fprintf(fp, "exit called @ statement fetch beginning: stmt #%d, \
%d blocks.\n",
            P->Vals.StmtFetchBegin.StatementNo,
            P->Vals.StmtFetchBegin.BlockCnt);
        break;

    case NXEventStmtFetchEnd :
        fprintf(fp, "exit called @ statement fetch end: %d records.\n",
            P->Vals.StmtFetchEnd.Records);
        break;

    case NXEventReqFetchEnd :
        fprintf(fp, "exit called @ request fetch ends: \
Records exported: %d, Records rejected: %d\n",
            P->Vals.ReqFetchEnd.RecsExported,
            P->Vals.ReqFetchEnd.RecsRejected);
        break;

    case NXEventExportEnd :
        fprintf(fp, "exit called @ export ends: \
Records exported: %d, Records rejected: %d\n",
            P->Vals.ExportEnd.RecsExported,
            P->Vals.ExportEnd.RecsRejected);
        break;
    }
    fclose(fp);
    return(0);
}

```





# User-Defined-Types and User-Defined-Methods

---

This appendix provides information on User-Defined-types (UDFs) and User-Defined-Methods (UDMs):

- [User-Defined-Types and User-Defined-Methods](#)
- [User-Defined-Methods \(UDMs\)](#)
- [Creating UDTs with FastExport](#)
- [Inserting and Retrieving UDTs with Client Products](#)
- [External Types](#)
- [Inserting UDTs with FastExport](#)
- [Retrieving UDTs with FastExport](#)
- [Retrieving UDT Metadata with FastExport](#)

## User-Defined-Types and User-Defined-Methods

This section provides a brief overview of how Teradata Database client products support user-defined custom data types known as User-Defined Types (UDTs), and user-defined custom functions known as User-Defined Methods (UDMs).

- User-Defined Types (UDTs)
- User-Defined Methods (UDMs)

For more in depth information on using UDTs and UDMs see, *Introduction to Teradata*, B035-1091-mmxA, *SQL Reference, UDF, UDM and External Stored Procedure Programming*, B035-1147-mmxA and *SQL Fundamentals*, B035-1141-mmxA.

### User-Defined Types (UDTs)

UDTs can be created to provide representations of real world entities within the Teradata Database. Choosing a set of UDTs which closely matches the entities encountered in a given business can yield a system which is easier to understand and hence easier to maintain. Support for UDTs and UDMs integrates the power of object-oriented technology directly into the Teradata Database.

## User-Defined-Methods (UDMs)

Teradata Database developer-created custom functions, which are explicitly connected to UDTs are known as User-Defined-Methods (UDMs). All UDMs must reside on server.

UDMs can be used to create an interface to the UDT that is independent of the UDT's internal representation. This makes it possible to later enhance a given UDT even in cases where the internal representation of the UDT must be changed to support the enhancement, without changing all of the database applications which use the UDT.

## Creating UDTs with FastExport

Teradata FastExport cannot create a custom UDT.

## Inserting and Retrieving UDTs with Client Products

A UDT can only exist on the Teradata Database server. Each UDT has an associated “from-sql routine” and “to-sql routine”.

- Insert - The “to-sql routine” constructs a UDT value from a pre-defined type value. The “to-sql routine” is automatically invoked when inserting values from a client system into a UDT on the Teradata Database server.
- Retrieve - The “from-sql routine” generates a pre-defined type value from a UDT. The “from-sql routine” is automatically invoked when a UDT is retrieved from the Teradata Database server to a client system.

## External Types

The “from-sql routine” and the “to-sql routine” create a mapping between a UDT and a pre-defined type. This pre-defined type is called the external type of a UDT. A client application only deals with the external type; it does not deal with UDT value directly.

### External Type Example

For example, if the following conditions exist:

- UDT named FULLNAME exists
- The external type associated with FULLNAME is VARCHAR(46)

Then, during an retrieve of FULLNAME values, the Teradata Database server converts the values from FULLNAME values to VARCHAR(46) values by invoking the “from-sql routine” associated with the FULLNAME UDT.

**Note:** The client must expect to receive the data in the same format as it receives VARCHAR(46) values.

Similarly, when values are provided by the client for insert into a FULLNAME UDT, the client must provide values in the same way it would provide values for a VARCHAR(46) field. The Teradata Database server will convert the values from VARCHAR(46) to FULLNAME values using the “to-sql routine” associated with the FULLNAME UDT.

## Inserting UDTs with FastExport

UDTs cannot be inserted with FastExport.

## Retrieving UDTs with FastExport

The Teradata FastExport utility can retrieve values from tables containing UDT columns in the same manner as is done for other tables.

If the select-list of the SELECT statement used in the Teradata FastExport job contains a UDT expression, the Teradata Database server automatically converts the UDT data to its external type before returning the data to the Teradata FastExport utility.

As such, the data written to the output location referenced by the “RETRIEVE” command will be in the external type associated with the UDTs.

## Retrieving UDT Metadata with FastExport

UDT Metadata cannot be retrieved with FastExport.



## Numeric

**24x7 Lights Out Operations:** The use of Systems Management tools to ensure the reliable movement and update of data from operational systems to analytical systems.

**2PC:** Two-Phase Commit

## 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.

**ABORT:** In *Teradata SQL*, a statement that stops a transaction in progress and backs out changes to the database only if the conditional expression associated with the abort statement is true.

**Access Lock:** A lock that allows selection of data from a table that may be locked for write access. The Teradata MultiLoad utility maintains access locks against the target tables during the Acquisition Phase.

**Access Module:** A software component that provides a standard set of I/O functions to access data on a specific device.

**Access Module Processor (AMP):** A virtual processor that receives steps from a parsing engine (PE) and performs database functions to retrieve or update data. Each AMP is associated with one virtual disk, where the data is stored. An AMP manages only its own virtual disk and not the virtual disk of any other AMP.

**access right:** A user's right to perform the *Teradata SQL* statements granted to him against a table, database, user, macro, or view. Also known as privilege.

**account:** The distinct account name portion of the system account strings, excluding the performance group designation. Accounts can be employed wherever a user object can be specified.

**Acquisition Lock:** A lock that is a flag in the table header that effectively rejects certain types of *Teradata SQL* access statements. An acquisition lock allows all concurrent DML access and the DROP DDL statement, and rejects DDL statements other than DROP.

**Acquisition Phase:** Responsible for populating the primary data subtables of the work tables. Data are received from the host, converted into internal format, and inserted into the work tables. The work tables will be sorted at the end of the Acquisition Phase and prior to the Application Phase.

**action definition:** A logical action consisting of a single physical action and related attributes.

**active data warehouse (ADW):** An active data warehouse provides information that enables decision-makers within an organization to manage customer relationships quickly, efficiently and proactively. Active data warehousing is about integrating advanced decision support with day-to-day, even minute-to-minute decision making that increases quality which encourages customer loyalty and thus secures an organization's bottom line. The market is maturing as it progresses from first-generation "passive" decision-support systems to current- and next-generation "active" data warehouse implementations.

**Active Database:** Active database systems integrate event-based rule processing with traditional database functionality. The behavior of the database is achieved through a set of Event-Condition-Action rules associated with the database. When an event is detected the relevant rules fire. Firing of a rule implies evaluating a condition on the database and carrying out the corresponding action. An active database system derives its power from the variety of events it can respond to and the kind of actions it can perform in response.

**Ad Hoc Query:** Any query that cannot be determined prior to the moment the query is issued.

**administrator:** A special user responsible for allocating resources to a community of users.

**Aggregation:** Used in the broad sense to mean aggregating data horizontally, vertically, and chronologically.

**all joins:** In *Teradata SQL*, a join is a SELECT operation that allows you to combine columns and rows from two or more tables to produce a result. Join types restricted by DWM are: inner join, outer join, merge join, product join, and all joins.

All joins are a combination of the above types, depending on how the user selects the information to be returned. In addition to the four types listed above, selecting all joins may include an exclusion join, nested join, and RowID join.

**allocation group:** (AG) A set of parameters that determine the amount of resources available to the sessions assigned to a PG referencing a specific AG. Has an assigned weight that is compared to other AG weights. An AG can limit the total amount of CPU used by sessions under its control.

**AMP:** Access Module Processor (UNIX-based systems), a type of virtual processor (vproc) that controls the management of the Teradata Database and the disk subsystem, with each AMP being assigned to a virtual disk (vdisk). For more information, see the *Introduction to Teradata*.

**AMP worker task:** (AWT) Processes (threads on some platforms) dedicated to servicing the Teradata Database work requests. For each AMP vproc, a fixed number of AWTs are pre-allocated during Teradata Database initialization. Each AWT looks for a work request to arrive in the Teradata Database, services the request, and then looks for another. An AWT can process requests of any work type. Each Teradata Database query is composed of a series of work requests that are performed by AWTs. Each work request is assigned a work type

indicating when the request should be executed relative to other work requests waiting to execute.

**Analytical Data Store:** Useful in making strategic decisions, this data storage area maintains summarized or historical data. This stored data is time variant, unlike operational systems which contain real-time data. Information contained in this data store is determined and collected based on the corporate business rules.

**ANSI:** American National Standards Institute. ANSI maintains a standard for SQL. For information about Teradata compliance with ANSI SQL, see the *SQL Fundamentals*.

**AP:** Application Processor

**APE:** Alert Policy Editor. Use this Teradata Manager component to define alert policies: create actions, set event thresholds, assign actions to events, and apply the policy to the Teradata Database.

**APH:** Alternate Parcel Header.

**Application Lock:** A flag set in the table header of a target table indicating that the Application Phase is in progress. An application lock allows all concurrent access lock select access and the DROP DDL statement, and rejects all other DML and DDL statements.

**Application Lifecycle:** Includes the following three stages:

- process and change management
- analysis and design
- construction and testing

**Application Phase:** Responsible for turning rows from a work table into updates, deletes, and inserts and applying them to a single target table.

**APRC:** Application Processor Reset Containment

**API:** Application Program Interface. An interface (calling conventions) by which an application program accesses an operating system and other services. An API is defined at source code level and provides a level of abstraction between the application and the kernel (or other privileged utilities) to ensure the portability of the code.

An API can also provide an interface between a high level language and lower level utilities and services written without consideration for the calling conventions supported by compiled languages. In this case, the API may translate the parameter lists from one format to another and the interpret call-by-value and call-by-reference arguments in one or both directions.

**Architecture:** A definition and preliminary design which describes the components of a solution and their interactions. An architecture is the blueprint by which implementers construct a solution which meets the users' needs.

**ASCII:** American Standard Code for Information Interchange, a character set used primarily on personal computers.

**Availability:** A measure of the percentage of time that a computer system is capable of supporting a user request. A system may be considered unavailable as a result of events such as system failures or unplanned application outages.

## B

**B Tree:** An indexing technique in which pointers to data are kept in a structure such that all referenced data is equally accessible in an equal time frame.

**BAR:** Backup and restore; also referred to as Backup/Archive/Restore; a software and hardware product set.

**BLOB:** An acronym for binary large object. A BLOB is a large database object that can be anything that doesn't require character set conversion. This includes MIDI, MP3, PDF, graphics and much more. BLOBs can be up to 2 GB in size.

**BTEQ:** Teradata Basic Teradata Query facility. A utility that allows users on a workstation to access data on a Teradata Database, and format reports for both print and screen output.

**Business-Driven:** An approach to identifying the data needed to support business activities, acquiring or capturing those data, and maintaining them in a data resource that is readily available.

**bypass objects:** Specific users, groups, and accounts can be set up to circumvent DWM query management by declaring them to be bypassed. Basically, this turns off the DWM query checking mechanism for all of the requests issued by those users and/or using those accounts.

## C

**Call-Level Interface Version 2 (CLIV2):** A collection of callable service routines that provide an interface to the Teradata Database. Specifically, CLI is the interface between the application program and the Micro Teradata Directory Program (for network-attached clients). CLI builds parcels that MTDP packages for sending to the Teradata Database using the Micro Operating System Interface (for network-attached clients), and provides the application with a pointer to each of the parcels returned from the Teradata Database.

**Capture:** The process of capturing a production data source.

**cardinality:** In set theory, cardinality refers to the number of members in the set. When specifically applied to database theory, the cardinality of a table refers to the number of rows contained in a table.

**Change Data Capture:** The process of capturing changes made to a production data source. Change data capture is typically performed by reading the source DBMS log. It consolidates units of work, ensures data is synchronized with the original source, and reduces data volume in a data warehousing environment.

**channel-attached:** A mainframe computer that communicates with a server (for example, a Teradata Database) through a channel driver.



**Character Set:** A grouping of alphanumeric and special characters used by computer systems to support different user languages and applications. Various character sets have been codified by the American National Standards Institute (ANSI).

**Checkpoint Rate:** The interval between checkpoint operations during the Acquisition Phase of a Teradata MultiLoad import task expressed as either the number of rows read from your client system or sent to the Teradata Database, or an amount of time, in minutes.

**CICS:** Customer Information Control System

**CLI:** Call-Level Interface. The interface between the application program and the MTDP (for network-attached clients) or TDP (for channel-attached clients). CLIV2 refers to version two of the interface.

**Client:** A computer that can access the Teradata Database.

**CLIV2:** Call-Level Interface Version 2. The interface between the application program and the MTDP (for network-attached clients) or TDP (for channel-attached clients).

**CLIV2so:** Call-Level Interface Version 2 Shared Object (CLIV2so); this program installs the CLI libraries required by other utilities. When the CLIV2so program submits a request to a Teradata Database, CLI Library components transform the request into Teradata Database formats. The CLI Library sends requests to, and receives responses from, the Teradata Database over a network.

**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. The server performs shared functions such as managing communication and providing database services. The client performs individual user functions such as providing customized interfaces, performing screen-to-screen navigation, and offering help functions.

**CMS:** Conventional Monitor System

**CLOB:** An acronym for character large object. A CLOB is a pure character-based large object in a database. It can be a large text file, HTML, RTF or other character-based file. CLOBs can be up to 2 GB in size. Also see BLOB and LOB.

**Cluster:** Logical, table-level archive whereby only those rows residing on specific AMPs, and which are members of the specified cluster, are archived onto a single tape data set. This allows multiple jobs to be applied for backup of large tables, to reduce the backup window. This method is used to affect a parallel archive/restore operation via a “divide and conquer” backup strategy.

**COBOL:** Common Business-Oriented Language

**Coexistence System:** A Teradata system running on mixed platforms

**column:** In the relational model of *Teradata SQL*, databases consist of one or more tables. In turn, each table consists of fields, organized into one or more columns by zero or more rows. All of the fields of a given column share the same attributes.

**consumer:** In Teradata Parallel Transporter (Teradata PT), a type of operator that accepts data from other operators and stores it in an external data store, such as a file, Teradata Database table, and so on. A consumer operator *consumes* the data from the data stream's buffer.

**COP:** Communications Processor. One kind of interface processor (IFP) on the Teradata Database. A COP contains a gateway process for communicating with workstations via a network.

**COP Interface:** Workstation-resident software and hardware, and Teradata Database-resident software and hardware, that allows workstations and the Teradata Database to communicate over networks.

**CPU:** Central processing unit.

## D

**DASD:** Direct access storage device (pronounced DAZ-dee). A general term for magnetic disk storage devices that has historically been used in the mainframe and minicomputer (mid-range computer) environments. When used, it may also include hard disk drives for personal computers. A recent form of DASD is the redundant array of independent disks (RAID).

The "direct access" means that all data can be accessed directly in about the same amount of time rather than having to progress sequentially through the data.

**database:** A related set of tables that share a common space allocation and owner. A collection of objects that provide a logical grouping for information. The objects include, tables, views, macros, triggers, and stored procedures.

**Data Cardinality:** Cardinality is a property of data elements which indicates the number of allowable entries in the element. A data element such as gender only allows two entries (male or female) and is said to possess low cardinality. Data elements for which many allowable entries are possible, such as age or income are said to have high cardinality.

**Data Definition Language (DDL):** In *Teradata SQL*, the statements and facilities that manipulate database structures (such as CREATE, MODIFY, DROP, GRANT, REVOKE, and GIVE) and the *Data Dictionary* information kept about those structures. In the typical, pre-relational data management system, data definition and data manipulation facilities are separated, and the data definition facilities are less flexible and more difficult to use than in a relational system.

**Data Connector operator:** A Teradata PT (producer- and consumer-type) operator that emulates the Data Connector API within the Teradata PT infrastructure.

**Data Dictionary:** In the Teradata Database, the information automatically maintained about all tables, views, macros, databases, and users known to the Teradata Database system, including information about ownership, space allocation, accounting, and access right relationships between those objects. *Data Dictionary* information is updated automatically during the processing of *Teradata SQL* data definition statements, and is used by the *parser* to obtain information needed to process all *Teradata SQL* statements.

**data loading:** The process of loading data from a client platform to a Teradata Database server.

**data manipulation:** In *Teradata SQL*, the statements and facilities that change the information content of the database. These statements include INSERT, UPDATE, and DELETE.

**Data Mart:** A type of data warehouse designed to meet the needs of a specific group of users such as a single department or part of an organization. Typically a data mart focuses on a single subject area such as sales data. Data marts may or may not be designed to fit into a broader enterprise data warehouse design.

**Data Mining:** A process of analyzing large amounts of data to identify hidden relationships, patterns, and associations.

**Data Model:** A logical map that represents the inherent properties of the data independent of software, hardware, or machine performance considerations. The model shows data elements grouped into records, as well as the association around those records.

**Data Synchronization:** The process of identifying active data replicates and ensuring that data concurrency is maintained. Also known as data version synchronization or data version concurrency because all replicated data values are consistent with the same version as the official data.

**Data Scrubbing:** The process of filtering, merging, decoding, and translating source data to create validated data for the data warehouse.

**data streams:** Buffers in memory for temporarily holding data. A data stream is not a physical file; instead, it is more like a pipe (in UNIX or Windows), or a batch pipe in z/OS.

**Data Warehouse:** A subject oriented, integrated, time-variant, non-volatile collection of data in support of management's decision making process. A repository of consistent historical data that can be easily accessed and manipulated for decision support.

**DB2:** IBM DATABASE 2

**DBA:** Database Administrator

**DBQL:** Database Query Log. DBQLs are a series of system tables created in the DBC database during the Teradata Database installation process. They are used to track query processing. See *Database Administration* to learn more about the DBQL.

**DD:** Data dictionary or data definition.

**DDL:** Data definition language, which supports manipulating database structures and the Data Dictionary information kept about these structures.

**DDL operator:** The DDL operator is a stand-alone operator that allows you to perform any necessary database routines prior to a load/apply job without having to use another utility such as BTEQ. For example, you can create tables or indexes, or drop tables, as needed, before starting a load/apply job. As a stand-alone operator, supporting only one instance, the DDL operator does not send or retrieve data to or from a Teradata PT operator interface.

**DEFINE Statement:** A statement preceding the INSERT statement that describes the fields in a record before the record is inserted in the table. This statement is similar to the SQL USING clause.

**Delete Task:** A task that uses a full file scan to remove a large number of rows from a single Teradata Database table. A delete task is composed of three major phases: Preliminary, Application, and End. The phases are a collection of one or more transactions that are processed in a predefined order according to the MLOAD protocol.

**delimiter:** In *Teradata SQL*, a punctuation mark or other special symbol that separates one clause in a *Teradata SQL* statement from another, or that separates one *Teradata SQL* statement from another.

**DIT:** Directory Information Tree. A graphical display of an organization's directory structure, sites, and servers, shown as a branching structure. The top-level (root) directory usually represents the organization level.

**DLL:** Dynamic-link library. A feature of the Windows family of operating systems that allows executable routines to be stored separately as files with *.dll* extensions and to be loaded only when needed by a program.

**DML:** Data manipulation language. In *Teradata SQL*, the statements and facilities that manipulate or change the information content of the database. These statements include SELECT, INSERT, UPDATE, and DELETE.

**domain name:** A group of computers whose host names (the unique name by which a computer is known on a network) share a common suffix, that is the domain name.

**Drill down:** A method of exploring detailed data that was used in creating a summary level of data.

**DSN:** Digital Switched Network. The completely digital version of the PSTN.

**Dual Active System:** A dual active system is comprised of two active database systems that operate in tandem and serve the needs of both the production and development environments. Dual active systems virtually eliminate all down time and provide seamless disaster recovery protection for critical users and applications.

**Duplicate Row Check:** A logic within the Teradata Database used to check for duplicate rows while processing each primary data row for INSERTs and UPDATEs.

**DWM:** Dynamic Workload Manager. The product described in this document, which manages access to the Teradata Database.

**EBCDIC:** Extended binary coded decimal interchange code. An IBM code that uses 8 bits to represent 256 possible characters. It is used primarily in IBM mainframes, whereas personal computers use ASCII.

**E-CLI:** Extended Call-Level Interface

**Error Tables:** Tables created during the Preliminary Phase used to store errors detected while processing a Teradata MultiLoad job. There are two error tables, ET and UV, that contains errors found during the Acquisition Phase and Application Phase, respectively.

**EOF:** End of File

**ETL:** Extract, transform, and load

**EUC:** Extended UNIX Code. Extended UNIX Code (EUC) for Japanese and Traditional-Chinese defines a set of encoding rules that can support from 1 to 4 character sets.

**exclusion join:** In Teradata SQL, a product join or merge join where only the rows that do not satisfy (are NOT in) the conditional specified in the SELECT are joined.

**Exclusive Lock:** Supports the manual recovery procedure when a RELEASE MLOAD statement is executed after a MultiLoad task has been suspended or aborted.

**execution time frame:** A period of time when DWM can execute scheduled requests that are waiting to run.

**Export operator:** A Teradata PT producer-type operator that emulates some of the functions of the FastExport utility in the Teradata PT infrastructure.

**Extract:** The process of copying a subset of data from a source to a target environment.

**Exit Routines:** Specifies a predefined action to be performed whenever certain significant events occur during a Teradata MultiLoad job.

## F

**Failover:** Failover is when Teradata QD switches from one connected system to another when an error occurs. Many factors affect how failover occurs.

**failure:** Any condition that precludes complete processing of a *Teradata SQL* statement. Any failure will abort the current transaction.

**FastExport:** Teradata FastExport utility. A program that quickly transfers large amounts of data from tables and views of the Teradata Database to a client-based application.

**FastExport OUTMOD Adapter operator:** A Teradata PT consumer-type operator that acts as a “wrapper” for Teradata FastExport utility OUTMOD routines, allowing you to use them within the Teradata PT infrastructure.

**FastLoad:** Teradata FastLoad utility. A program that loads empty tables on the Teradata Database with data from a network-attached or channel-attached client.

**FastLoad INMOD Adapter operator:** A Teradata PT producer-type operator that acts as a “wrapper” for Teradata FastLoad utility INMOD routines, allowing you to use them within the Teradata PT infrastructure.

**field:** The basic unit of information stored in the Teradata Database. A field is either null, or has a single numeric or string value. See also *column*, *database*, *row*, *table*.

**FIFO:** First In first out queue.

**FIPS:** Federal Information Processing Standards

**filter operator:** In Teradata PT, a type of operator that performs filtering on data en route from other operators.

**Flat File** As a noun, an ASCII text file consisting of records of a single type, in which there is no embedded structure information governing relationships between records.

As an adjective, describes a flattened representation of a database as single file from which the structure could implicitly be rebuilt.

A particular type of database structure, as opposed to relational.

**Foreign Key:** The primary key of a parent data subject that is placed in a subordinate data subject. Its value identifies the data occurrence in the parent data subject that is the parent of the data occurrence in the subordinate data subject.

**Formatted Records:** See Records.

**Function:** User Defined Functions (UDF) are extensions to *Teradata SQL*. Users can write UDFs to analyze and transform data already stored in their data warehouse in ways that are beyond the functionality of Teradata's native functions.

## G

**Gateway:** A device that connects networks having different protocols.

**global rule:** Object Access and Query Resource rules can be specified as being global, that is, they apply to all objects, and therefore to all requests. When a rule is specified as being global, no query objects need be (or can be) associated with the rule because all objects are implicitly included. Care should be taken defining a global access rule, as it causes all requests to be rejected except those from the DBC user and any bypassed objects.

**Globally Distributed Objects (GDO):** A data structure that is shared by all of the virtual processors in the Teradata Database system configuration.

**graphical user interface (GUI):** The use of pictures rather than just words to represent the input and output of a program. A program with a GUI runs under a Windows operating system. The GUI displays certain icons, buttons, dialog boxes in its windows on the screen and the user controls it by moving a pointer on the screen (typically controlled by a mouse) and selecting certain objects by pressing buttons on the mouse. This contrasts with a command line interface where communication is by exchange of strings of text.

**GSS:** Generic Security Services. An application level interface (API) to system security services. It provides a generic interface to services which may be provided by a variety of different security mechanisms. Vanilla GSS-API supports security contexts between two entities (known as "principals").

## H

**heuristics:** Statistics recommendations, based on general rules of thumb.

**HOSI:** Acronym for hash-ordered secondary index.

## I

**IPT:** I/Os Per Transaction

**import:** This refers to the process of pulling system information into a program. To add system information from an external source to another system. The system receiving the data must support the internal format or structure of the data.

**Import Task:** A task that quickly applies large amounts of client data to one or more tables or views on the Teradata Database. Composed of four major phases: Preliminary, Acquisition, Application, and End. The phases are a collection of one or more transactions that are processed in a predefined order according to the MLOAD protocol. An import task references up to five target tables.

**In-Doubt:** A transaction that was in process on two or more independent computer processing systems when an interruption of service occurred on one or more of the systems. The transaction is said to be in doubt because it is not known whether the transaction was successfully processed on all of the systems.

**Information engineering:** The discipline for identifying information needs and developing information systems that produce messages that provide information to a recipient. Information engineering is a filtering process that reduces masses of data to a message that provides information.

**INMOD:** *Input Module*, a program that administrators can develop to select, validate, and preprocess input data.

**INMOD Routine:** User-written routines that MultiLoad and other load/export utilities use to provide enhanced processing functions on input records before they are sent to the Teradata Database. Routines can be written in C language (for network-attached platforms), or SAS/S, COBOL, PL/I or Assembler (for channel-attached platforms). A routine can read and preprocess records from a file, generate data records, read data from other database systems, validate data records, and convert data record fields.

**inner join:** In Teradata SQL, a join operation on two or more tables, according to a join condition, that returns the qualifying rows from each table.

**instance:** In object-oriented programming, refers to the relationship between an object and its class. The object is an instance of the class. In Teradata PT, an instance is an occurrence of a fully defined Teradata PT operator, with its source and target data flows, number of sessions, etc. Teradata PT can process multiple instances of operators.

**interface processor (IFP):** Used to manage the dialog between the Teradata Database and the host. Its components consist of session control, client interface, the parser, the dispatcher,

and the BYNET. One type of IFP is a communications processor (COP). A COP contains a gateway process for communicating with workstations via a network.

**Intermediary:** A computer software process written by a third party which interfaces to one or more Teradata servers and initiates a change data capture or change data apply operation with replication services.

**internet protocol (IP):** Data transmission standard; the standard that controls the routing and structure of data transmitted over the Internet.

**interval histogram:** Interval histograms are a form of synopsis data structure. A synopsis data structure is a data structure that is substantially smaller than the base data it represents. Interval histograms provide a useful statistical profile of attribute values that characterize the properties of that raw data. The Teradata Database uses interval histograms to represent the cardinalities and certain other statistical values and demographics of columns and indexes for all-AMPs sampled statistics and for full-table statistics. Each histogram is composed of a maximum of 100 intervals.

**I/O:** Input/output.

**ISO:** International Standards Organization

## J

**JIS:** Japanese Industrial Standards specify the standards used for industrial activities in Japan. The standardization process is coordinated by Japanese Industrial Standards Committee and published through Japanese Standards Association.

**Job Script:** A job script, or *program*, is a set of MultiLoad commands and Teradata SQL statements that make changes to specified target tables and views in the Teradata Database. These changes can include inserting new rows, updating the contents of existing rows, and deleting existing rows.

**join:** A select operation that combines information from two or more tables to produce a result.

## L

**LAN:** Local Area Network. LANs supported by Teradata products must conform to the IEEE 802.3 standard (Ethernet LAN).

**Least Used:** Lease used (*-lu*) in a command line parameter that tells Teradata QD to route queries to the least used database.

**Load operator:** A Teradata PT consumer-type operator that emulates some of the functions of the FastLoad utility in the Teradata PT infrastructure.

**LOB:** An acronym for large object. A large object is a database object that is large in size. LOBs can be up to 2 gigabytes. There are two types of LOBs, CLOBs and BLOBs. CLOBs are character-based objects, BLOBs are binary-based objects.



**Locks:** FastLoad automatically locks any table being loaded and frees a lock only after an END LOADING statement is entered. Therefore, access to a table is available when FastLoad completes.

**log:** A record of events. A file that records events. Many programs produce log files. Often you will look at a log file to determine what is happening when problems occur. Log files have the extension “.log”.

**log stream:** A log stream is a series of log messages defined in one message catalog and initiated from one originator. One originator may initiate several log streams (for example, if there are multiple operators in one originator).

**logical action:** A named action that is defined on the Alert Policy Editor's Actions tab. Logical actions can be assigned to events in the alert policy.

**Logical Data Model:** A data model that represents the normalized design of data needed to support an information system. Data are drawn from the common data model and normalized to support the design of a specific information system.

Actual implementation of a conceptual module in a database. It may take multiple logical data models to implement one conceptual data model.

**loner value:** A value that has a frequency greater than the total number of table rows divided by the maximum interval times 2.

## M

**MAPI:** Messaging Application Programming Interface. A set of Microsoft-defined functions and interfaces that support E-mail capabilities.

**macro:** a file that is created and stored on the Teradata Database, and is executed in response to a *Teradata SQL EXECUTE* statement

**merge join:** In *Teradata SQL*, the type of join that occurs when the WHERE conditional of a SELECT statement causes the system first to sort the rows of two tables based on a join field (specified in the statement), then traverse the result while performing a merge/match process.

**Metadata:** Data about data. For example, information about where the data is stored, who is responsible for maintaining the data, and how often the data is refreshed.

**methods:** In object-oriented programming, methods are the programming routines by which objects are manipulated.

**NFS:** Network file system.

**MIB:** Management Information Base

**MOSI:** Micro Operating System Interface. A library of routines that implement operating system dependent and protocol dependent operations on the workstation.

**MTDP:** Micro Teradata Director Program. A library of routines that implement the session layer on the workstation. MTDP is the interface between CLI and the Teradata Database.

**MPP:** Massively Parallel Processing

**multi-threading:** An option that enables you to speed up your export and import operations with multiple connections.

**MultiLoad:** Teradata MultiLoad. A command-driven utility that performs fast, high-volume maintenance functions on multiple tables and views of the Teradata Database.

**Multiset Tables:** Tables that allow duplicate rows.

**MVS (Multiple Virtual Storage):** One of the primary operating systems for large IBM computers.

## N

**name:** A word supplied by the user that refers to an object, such as a *column, database, macro, table, user, or view*.

**nested join:** In *Teradata SQL*, this join occurs when the user specifies a field that is a unique primary index on one table and which is in itself an index (unique/non-unique primary or secondary) to the second table.

**Network:** In the context of the Teradata Database, a LAN (*see* LAN).

**network attached:** A computer that communicates over the LAN with a server (for example, a Teradata Database).

**NIC:** Network Interface Card.

**NO REWIND:** A tape device definition that prevents a rewind operation at either file open or file close. NO REWIND allows a program to access multiple files on a tape by leaving the tape positioned at the end of the current file at close, thus allowing the subsequent file to be easily accessed by the next open.

**notify exit:** A user-defined exit routine that specifies a predefined action to be performed whenever certain significant events occur during a Teradata PT job.

For example, by writing an exit in C (without using CLIV2) and using the NotifyExit attribute in an operator definition, you can provide a routine to detect whether a Teradata PT job succeeds or fails, how many records were loaded, what the return code is for a failed job, and so on.

**null:** The absence of a value for a field.

**Nullif Option:** This option allows the user to null a column in a table under certain conditions; it is only used in conjunction with DEFINE statements.

**NUPI:** Non-unique primary index; an NUPI is typically assigned to minor entities in the database.

**NUSI:** Non-unique secondary index; an NUSI is efficient for range query access, while a unique secondary index (USI) is efficient for accessing a single value.

## O

**object:** In object-oriented programming, a unique instance of a data structure defined according to the template provided by its class. Each object has its own values for the variables belonging to its class and can respond to the messages, or methods, defined by its class.

**object access rule:** An Object Access filter allows you to define the criteria for limiting access to issuing objects and/or query objects. Queries that reference objects associated with the rule (either individually or in combination) during the specified dates and times are rejected. Global rules are not applicable for this type.

**object definition:** The details of the structure and instances of the objects used by a given query. Object definitions are used to create the tables, views, and macros, triggers, join indexes, and stored procedures in a database.

**ODBC:** (Open Database Connectivity) Under ODBC, drivers are used to connect applications with databases. The ODBC driver processes ODBC calls from an application, but passes SQL requests to the Teradata Database for processing.

**ODBC operator:** A Teradata PT producer-type operator that enables universal open data access with many ODBC-compliant data sources, including Oracle, SQL Server, DB2, and so on. The ODBC operator runs on all Teradata PT supported platforms. It reads data close to the sources, and then feeds the data directly to the Teradata Database without the need of an intermediate staging platform.

**OLTP:** (On-Line Transaction Processing) Processing that supports the daily business operations. Also known as operational processing.

**operator routine:** In object-oriented programming, refers to a function that implements a method.

The terms *operator routine* and *operator function* may be used interchangeably.

**OS/VS** Operating System/Virtual Storage

**OTB:** Open Teradata Backup; a product set consisting of OTB-Veritas, OTB-BakBone, and others; Teradata backup products for MP-RAS/UNIX, NT and Windows platforms.

**outer join:** In *Teradata SQL*, an extension of an inner join operation. In addition to returning qualifying rows from tables joined according to a join condition (the inner join), an outer join returns non-matching rows from one or both of its tables. Multiple tables are joined two at a time.

**owner:** In *Teradata SQL*, the user who has the ability to grant or revoke all access rights on a database to and from other users. By default, the creator of the database is the owner, but ownership can be transferred from one user to another by the GIVE statement.

## P

**parameter:** A variable name in a macro for which an argument value is substituted when the macro is executed.

**parser:** A program executing in a *PE* that translates *Teradata SQL* statements entered by a user into the steps that accomplish the user's intentions.

**parsing engine (PE):** An instance (virtual processor) of the database management session control, parsing, and dispatching processes and their data context (caches).

**Paused MultiLoad Job:** A job that was halted, before completing, during the Acquisition Phase of the Teradata MultiLoad operation. The paused condition can be intentional, or the result of a system failure or error condition.

**PDE:** Parallel Database Extensions

**peak perm:** Highest amount of permanent disk space, in bytes, used by a table.

**performance groups:** A performance group is a collection of parameters used to control and prioritize resource allocation for a particular set of Teradata Database sessions within the Priority Scheduler. Every Teradata Database session is assigned to a performance group during the logon process. Performance groups are the primary consideration in partitioning the working capacity of the Teradata Database. To learn more about performance groups, see the Priority Scheduler section of *Utilities*.

**performance periods:** A threshold or limit value that determines when a session is under the control of that performance period. A performance period links PGs/Teradata Database sessions under its control to an AG that defines a scheduling strategy. A performance period allows you to change AG assignments based on time-of-day or resource usage.

**Physical Data Model:** A data model that represents the denormalized physical implementation of data that support an information system. The logical data model is denormalized to a physical data model according to specific criteria that do not compromise the logical data model but allow the database to operate efficiently in a specific operating environment.

**Pipeclient:** A command line program used to send commands to Teradata Query Director. The program uses named pipes formatting.

**Primary server:** A Teradata server in which client applications execute transactions through use of *Teradata SQL* or utilities such as Teradata MultiLoad and update the tables of one or more replication groups. The changes are captured by replication services and given to an intermediary connected to the server.

**priority definition set:** A collection of data that includes the resource partition, performance group, allocation group, performance period type, and other definitions that control how the Priority Scheduler manages and schedules session execution.

**product join:** In *Teradata SQL*, the type of join that occurs when the WHERE conditional of a SELECT statement causes the Teradata Database system to compare all qualifying rows from one table to all qualifying rows from the other table. Because each row of one table is compared to each row of another table, this join can be costly in terms of system performance.

Note that product joins without an overall WHERE constraint are considered unconstrained (Cartesian). If the tables to be joined are small, the effect of an unconstrained join on

performance may be negligible, but if they are large, there may be a severe negative effect on system performance.

**profiles:** A profile is a set of parameters you assign to a user, group of users, or an account that determines what scheduling capabilities are available and how your Teradata Query Scheduler scheduled requests server handles their scheduled requests.

**physical action:** A basic action type, such as <Send a Page>, <Send an E-Mail>, and so on. Physical actions must be encapsulated by logical actions in order to be used in the alert policy.

**PIC:** Position independent code

**PIPC:** Parallel InterProcess Communication. It is a software component that provides messaging protocols and parallelization primitives for Teradata PT to launch and execute parallel applications across the enterprise.

**Pipeclient:** A command line program used to send commands to Teradata Query Director. The program uses named pipes formatting.

**PL/I:** Programming Language/1, a programming language supported for MultiLoad development.

**PMPC:** Performance Monitor and Production Controls

**PP2:** Preprocessor2

**PPP:** Point-to-Point Protocol

**Primary Key:** A set of one or more data characteristics whose value uniquely identifies each data occurrence in a data subject. A primary key is also known as a unique identifier.

**privilege:** A user's right to perform the Teradata SQL statements granted to him against a table, database, user, macro, or view. Also known as access right.

**procedure:** Short name for Teradata stored procedure. Teradata provides Stored Procedural Language (SPL) to create stored procedures. A stored procedure contains SQL to access data from within Teradata and SPL to control the execution of the SQL.

**production system:** A database used in a live environment. A system that is actively used for day-to-day business operations. This differs from a test or development system that is used to create new queries or test new features before using them on the production system.

**Protocol:** The rules for the format, sequence and relative timing of messages exchanged on a network.

## Q

**query analysis:** A feature that estimates the answer set size (number of rows) and processing time of a **SELECT** type query.

**Query Capture Database (QCD):** A database of relational tables that store the steps of any query plan captured by the Query Capture Facility (QCF).

**Query Capture Facility (QCF):** Provides a method to capture and store the steps from any query plan in a set of predefined relational tables called the Query Capture Database (QCD).

**query:** A *Teradata SQL* statement, particularly a `SELECT` statement.

**Query Director:** A Teradata client application used to balance sessions between systems according to user provided algorithms.

**query management:** The primary function of DWM is to manage logons and queries. This feature examines logon and query requests before they are dispatched for execution within the Teradata Database, and may reject logons, and may reject or delay queries. It does this by comparing the objects referenced in the requests to the types of DBA-defined rules.

**Query Resource filter:** A Query Resource filter allows you to define the criteria for limiting resource usage associated with queries. You can define resource criteria such as:

- Row count
- Processing time
- No joins permitted
- No full table scans permitted

Queries that are estimated to meet or exceed the limits for the rule during the specified dates and times are rejected. You may define global rules for this type.

**Query Session Utility:** A separate utility program used to monitor the progress of your MultiLoad job. It reports different sets of status information for each phase of your job.

## R

**random AMP sample (RAS):** An arbitrary sample from an Access Module Processor (AMP). These are samples of the tables in a query or all of the tables in a given database. Also known as RAS.

**RDBMS (Relational Database Management System):** A database management system in which complex data structures are represented as simple two-dimensional tables consisting of columns and rows. For Teradata SET, RDBMS is referred to as “Teradata Database.”

**Records:** When using the Teradata MultiLoad utility, both formatted and unformatted records are accepted for loading. A formatted record, in the Teradata Database world, 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 the Teradata Database. Unformatted records are any records not originating on a Teradata Database, such as Lotus 1-2-3 files. These files contain records that must be defined before loading onto the Teradata Database.

**recursive query:** A named query expression that is allowed to reference itself in its own definition, giving the user a simple way to specify a search of a table using iterative self-join and set operations. Use a recursive query to query hierarchies of data. Hierarchical data could be organizational structures such as department and sub-department, forums of discussions such as posting, response, and response to response, bill of materials, and document hierarchies.

**Replication Group:** A set of tables for which either data changes are being captured on a primary server or applied on a subscriber server.

**Replication Services:** a set of software functions implemented in the Teradata server that interact with an intermediary to capture or apply change data to the tables of a replication group.

**request:** In host software, a message sent from an application program to the Teradata Database.

**resource partition:** A collection of prioritized PGs related by their users' associations. Has an assigned weight that determines the proportion of resources available to that partition relative to the other partitions defined for that Teradata Database.

**Restart Log Table:** One of four restart tables the Teradata MultiLoad utility creates that are required for restarting a paused Teradata MultiLoad job.

**Restoration Lock:** A flag set in the table header of a target table indicating that the table was aborted during the Application Phase and is now ready to be restored. A limited set of operations can be done on the table: Delete All, Drop Fallback, Drop Index, Drop Table, and Select with access lock. No Teradata MultiLoad restart will be allowed on a table with a Restoration Lock.

**result:** The information returned to the user to satisfy a request made of the Teradata Database.

**results table/file:** In the Schedule Request environment, a results table or file is a database table or a Windows file into which result data for a schedule request that is not self-contained are stored.

**results file storage:** A symbolic name to a root directory where scheduled requests results are stored. You map a file storage location to a Windows root directory where results are stored.

**RowID join:** In *Teradata SQL*, this join occurs when one of the join tables has a non-unique primary index constant, and another column of that table matches weakly with a non-unique secondary index column of the second table.

**rule:** Rules are the name given to the method used by DWM to define what requests are prohibited from being immediately executed on the Teradata Database. That is, the rules enforced by DWM provide the Query Management capabilities.

**row:** Whether null or not, that represent one entry under each *column* in a *table*. The row is the smallest unit of information operated on by data manipulation statements.

**RSG:** Relay Services Gateway. A virtual processor residing on a node in which the replication services software will execute.

**RT:** Response Time

**RTF:** Rich Text File

**run file:** A script that is not contained within the SYSIN file, but rather executed through use of the .RUN BTEQ command.

## S

**scheduled requests:** The capability to store scripts of SQL requests and execute them at a scheduled later time.

**schema:** Schemas are used to identify the structure of the data. Producers have an output schema, to define what the source data will look like in the data stream. Consumers have an input schema, to define what will be read from the data stream. If the input and output schemas are the same, you only define the schema once.

**script:** A file that contains a set of BTEQ commands and/or SQL statements.

**Security token:** A binary string generated by a server when a replication group is created or altered that must be input to secure a change data capture or apply operation.

**separator:** A character or group of characters that separates words and special symbols in *Teradata SQL*. Blanks and comments are the most common separators.

**server:** A computer system running the Teradata Database. Typically, a Teradata Database server has multiple nodes, which may include both TPA and non-TPA nodes. All nodes of the server are connected via the Teradata BYNET or other similar interconnect.

**session:** In client software, a logical connection between an application program on a host and the Teradata Database that permits the application program to send one request to and receive one response from the Teradata Database at a time.

**skew:** This value (which is only available in V2R4 and above) is calculated based on a single Database collection interval. If the Session Collection rate is 60, then the skew is calculated for a 60-second period.

The value is calculated using 'current' data values. For example, the Max CPU used during the past 60 seconds relative to the Average used over that same 60 seconds:

$$\text{skew} = 100 * (1 - \text{avg} / \text{max})$$

**SMP:** Symmetric Multi-Processing

**SNMP:** Simple Network Management Protocol. See the SNMP FAQ: <http://www.faqs.org/faqs/snmp-faq/>

**Sockclient:** A command line program used to send commands to Teradata Query Director.

**Source Database:** The database from which data will be extracted or copied into the Data Warehouse.

**SQL:** Structured Query Language. An industry-standard language for creating, updating and, querying relational database management systems. SQL was developed by IBM in the 1970s for use in System R. It is the *de facto* standard as well as being an ISO and ANSI standard. It is often embedded in general purpose programming languages.



Programming language used to communicate with the Teradata Database.

**SSO:** Single sign-on, an authentication option that allows users of the Teradata Database on Windows 2000 systems to access the Teradata Database based on their authorized network usernames and passwords. This feature simplifies the procedure requiring users to enter an additional username and password when logging on to Teradata Database via client applications.

**Star Schema:** A modeling scheme that has a single object in the middle connected to a number of objects around it radially.

**statement:** A request for processing by the Teradata Database that consists of a keyword verb, optional phrases, operands and is processed as a single entity.

**statistics:** These are the details of the processes used to collect, analyze, and transform the database objects used by a given query.

**stored procedure:** Teradata Version 2 Release 4 and later supports stored procedures. A stored procedure is a combination of SQL statements and control and conditional handling statements that run using a single call statement.

**Subscriber server:** A Teradata server in which changes captured from a primary server by an intermediary are applied to tables that duplicate those of the primary. Replication services executing on the servers provide the capture and apply functions.

**supervisory user:** In *Data Dictionary*, a *user* who has been delegated authority by the administrator to further allocate Teradata Database resources such as space and the ability to create, drop, and modify users within the overall user community.

## T

**table:** A set of one or more columns with zero or more rows that consist of fields of related information.

**Target Database:** The database in which data will be loaded or inserted.

**Target table:** A user table where changes are to be made by an MLOAD task.

**TCP/IP:** Transmission Control Protocol/Internet Protocol.

**TDDSMC:** Teradata Database System Management Console that allows users to view and perform maintenance activities on ARCMAN backups that are stored in Tivoli Storage Management.

**TDPID:** Teradata Director Program Identifier. The name of the Teradata Database being accessed.

**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).

**TDP:** Teradata Director Program; TDP provides a high-performance interface for messages communicated between the client and the Teradata system.

**Target Level Emulation (TLE):** Permits you to emulate a target environment (target system) by capturing system-level information from that environment. The captured information is stored in the relational tables *SystemFE.Opt\_Cost\_Table* and *SystemFE.Opt\_RAS\_Table*. The information in these tables can be used on a test system with the appropriate column and indexes to make the Optimizer generate query plans as if it were operating in the target system rather than the test system.

**test system:** A Teradata Database where you want to import Optimizer-specific information to emulate a target system and create new queries or test new features.

**title:** In *Teradata SQL*, a string used as a column heading in a report. By default, it is the column name, but a title can also be explicitly declared by a `TITLE` phrase.

**TPA:** Trusted Parallel Application.

**TOS:** Teradata Operating System

**TPM:** Transactions Per Minute

**Transport:** The process of extracting data from a source, interfacing with a destination environment, and then loading data to the destination.

**transaction:** A set of *Teradata SQL* statements that is performed as a unit. Either all of the statements are executed normally or else any changes made during the transaction are backed out and the remainder of the statements in the transaction are not executed. The Teradata Database supports both ANSI and Teradata transaction semantics.

**trigger:** One or more *Teradata SQL* statements associated with a table and executed when specified conditions are met.

**TSM:** Tivoli Storage Management; IBM's storage management solution.

**TTU:** Teradata Tools and Utilities is a robust suite of tools and utilities that enables users and system administrators to enjoy optimal response time and system manageability with their Teradata system. Teradata Fast Export is included in Teradata Tools and Utilities.

**tuple:** In a database table (relation), a set of related values one for each attribute (column). A tuple is stored as a row in a relational database management system. It is analogous to a record in a non relational file.

**Two Phase Commit:** Two Phase Commit is the process by which a relational database ensures that distributed transactions are performed in an orderly manner. In this system, transactions may be terminated by either committing them or rolling them back.

**type:** An attribute of a column that specifies the representation of data values for fields in that column. *Teradata SQL* data types include numerics and strings.

## U

**UDF** User Defined Functions

**UDM** User-Defined Methods. The database developer can create custom functions that are explicitly connected to UDTs; these are known as UDMs. Functionalities directly applicable to a UDT can be located within the UDMs associated with that UDT rather than being replicated to all of the applications that use that UDT, resulting in increased maintainability.

**UDT** A custom data type, known as a user-defined type. By creating UDTs, a database developer can augment the Teradata Database with data types having capabilities not offered by Teradata predefined (built-in) data types. Use Teradata FastExport to export values from tables containing UDT columns in the same manner as is done for other tables. If the *select-list* of the SELECT statement used in the Teradata FastExport job contains a UDT expression, the Teradata Database server automatically converts the UDT data to its external type before returning the data to the Teradata FastExport utility.

**Unformatted Records:** See Records.

**Unicode:** A fixed-width (16 bits) encoding of virtually all characters present in all languages in the world.

**unique secondary index (USI):** One of two types of secondary indexes. A secondary index may be specified at table creation or at any time during the life of the table. It may consist of up to 16 columns. To get the benefit of the index, the query has to specify a value for all columns in the secondary index. A USI has two purposes: It can speed up access to a row which otherwise might require a full table scan without having to rely on the primary index, and it can be used to enforce uniqueness of a column or set of columns.

**user:** In *Teradata SQL*, a database associated with a person who uses the Teradata Database. The database stores the person's private information and accesses other Teradata Databases.

**Update operator:** A Teradata PT consumer-type operator that emulates some of the functions of the Teradata MultiLoad utility in the Teradata PT infrastructure.

**UPI:** Unique primary index; a UPI is required and is typically assigned to major entities in the database.

**user:** A database associated with a person who uses the Teradata Database. The database stores the person's private information and accesses other Teradata Databases.

**user groups:** A group of users can be specified within DWM as either as a collection of individual users, or as all user names which satisfy a character string pattern (such as SALE\*). The Teradata concept of roles is not used to define user groups, as it applies to privileges. User groups can generally be employed wherever an issuing object can be specified, and any condition applied to a group implicitly applies to all users within that group.

**UTF-8:** In simple terms, UTF-8 is an 8 bit encoding of 16 bit Unicode to achieve an international character representation.

In more technical terms, in UTF-8, characters are encoded using sequences of 1 to 6 octets. The only octet of a sequence of one has the higher-order bit set to 0, the remaining 7 bits are

used to encode the character value. UTF-8 uses all bits of an octet, but has the quality of preserving the full US-ASCII range. The UTF-8 encoding of Unicode and UCS avoids the problems of fixed-length Unicode encodings because an ASCII file encoded in UTF is exactly same as the original ASCII file and all non-ASCII characters are guaranteed to have the most significant bit set (bit 0x80). This means that normal tools for text searching work as expected.

**UTF16** A 16-bit Unicode Translation Format.

## V

**value-ordered secondary index (VOSI):** A non-unique secondary index (NUSI) can be value ordered which means the NUSI can be sorted on the key values themselves rather than on the corresponding hash codes. This is useful for range queries where only a portion of the index subtable will be accessed. With a value-ordered NUSI, only those blocks in the NUSI subtable that are within the range are scanned. It must be a number value, up to 4 bytes, versus a longer character column. DATE is the most commonly used data type. The actual data value is stored as part of the NUSI structure.

**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.

**view:** An alternate way of organizing and presenting information in a Teradata Database. A view, like a table, has rows and columns. However, the rows and columns of a view are not directly stored by the Teradata Database. They are derived from the rows and columns of tables (or other views) whenever the view is referenced.

**VM (Virtual Machine):** One of the primary operating systems for large IBM computers.

**VM/CMS** Virtual Machine/Conversational Monitor System

## W

**workgroups:** Workgroups represent collections of related scheduled request work for users, user groups, or accounts. Each workgroup is assigned a maximum number of requests that can be executing from that workgroup simultaneously thereby ensuring that requests for all workgroups get a fair share of their scheduled work done within the execution time frames.

**workload limits rule** A Workload Limits rule allows you to limit the number of logon sessions and all-AMP queries, as well as reject or delay queries when workload limits are encountered. You can define which users, accounts, performance groups, or users within performance groups that are associated with this type of rule.

**Workstation:** A network-attached client.

**Work Table:** A table created during the Preliminary Phase used to store intermediate data acquired from the host during a Teradata MultiLoad task. These data will eventually be applied to a target table.

**Write Lock:** A write lock enables a single user to modify a table. The Teradata MultiLoad utility maintains write locks against each target table during the Application Phase, and work tables and error tables for each task transaction.

## X

**XML:** XML is the eXtensible Markup Language -- a system created to define other markup languages. For this reason, it can also be referred to as a metalanguage. XML is commonly used on the Internet to create simple methods for the exchange of data among diverse clients.

## Z

**z/OS (MVS (Multiple Virtual Storage)):** One of the primary operating systems for large IBM computers.

**z/VM (VM Virtual Machine and VM/CMS):** One of the primary operating systems for large IBM computers. Virtual Machine/Conversational Monitor System.



## Symbols

- & (ampersand) character, in variable substitution 41
- \* (asterisk character) as the fileid specification
  - ACCEPT command 65
  - DISPLAY command 76
  - ROUTE MESSAGES command 117
  - RUN FILE command 119
- ./ prefix
  - EXIT name specification, BEGIN EXPORT command 69
  - INMOD modulename specification, IMPORT command 100
  - OUTMOD modulename specification, EXPORT command 79
- @ character in OUTFILE fileid specifications 82

## Numerics

- 2633 error message 48

## A

- abort 31, 181
- ACCEPT command
  - defined 16
  - syntax 64
- Access Module, name specification 79, 100
- accounts
  - defined 181
- acctid specification, LOGON command 111
- acronyms 181
- administrator, defined 182
- all joins
  - defined 182
- ALTER TABLE statement, Teradata SQL 18
- alternate error file run-time parameter 24
- ANSI/SQL DateTime data types
  - programming considerations 40
  - specifications, table of 89
- ANSIDATE specification, DATEFORM command 74
- API
  - defined 183
- apostrophes, displaying in the text string, DISPLAY command 75
- ASCII
  - character set code 41
- Assembler language INMOD routines
  - programming structure 51

- AXSMOD
  - character set 44
  - option, EXPORT command 79
  - option, IMPORT command 100

## B

- b run-time parameter 23
- BEGIN EXPORT command
  - defined 17
  - syntax 67
- BINARY format specification
  - EXPORT command 80
- block size specifications, table of 83
- BLOCKSIZE integer specification, EXPORT command 80
- BRIEF
  - configuration file parameter 34, 35
  - run-time parameter 23
- bypass objects
  - defined 184

## C

- C language
  - INMOD examples for UNIX 156
  - INMOD routine programming structure 51
  - notify exit parameters 158
- character sets
  - AXSMOD 44
  - client system specifications 44
  - default 44
  - programming considerations 41
  - run-time parameters 44
  - supported 41
  - Teradata Database default 44
  - Unicode 26, 43
  - UTF16 26, 43
  - UTF8 26, 43
- charpos specification
  - ACCEPT command 64
  - RUN FILE command 118
- CHARSET configuration file parameter 34, 35
- CHECKPOINT statement, Teradata SQL 18
- CLIV2 error on your client system 33
- COBOL language INMOD routine programming structure 51
- COLLECT STATISTICS statement, Teradata SQL 18 command

- conventions 38
    - conditional expressions 38
    - operators 38
    - reserved words 39
  - commands
    - syntax, see individual commands
  - COMMENT statement, Teradata SQL 18
  - comments
    - programming considerations 40
    - using variable substitutions 41
    - using with Teradata SQL statements 41
  - compiling routine
    - HP-UX 162
    - IBM-AIX 164
    - LINUX 165
    - MP-RAS 161
    - SOLARIS 161
  - concurrent load utility tasks, programming considerations 48
  - conditional expressions 38
  - configuration file
    - and errors 36
    - contents 35
    - file name and location 35
    - optional specification 21
    - overriding internal utility defaults 35
    - parameters overridden by run-time parameters
      - BRIEF 34
      - CHARSET 24, 34
      - ERRLOG 24, 34
      - MAXSESS 24, 34
      - MINSESS 24, 34
    - processing 36
    - using 34
  - CONTINUEIF specification
    - LAYOUT command 105
    - with multibyte character sets 45
  - CREATE DATABASE statement, Teradata SQL 18
  - CREATE MACRO statement, Teradata SQL 18
  - CREATE TABLE statement, Teradata SQL 18
  - CREATE VIEW statement, Teradata SQL 18
- D**
- data
    - file concatenation, programming considerations 48
    - type specification, changing
      - SET command 120
  - data dictionary, defined 186
  - data encryption
    - run-time option specification 28
  - data manipulation, defined 187
  - DATABASE statement, Teradata SQL 18
  - datadesc specification
    - FIELD command 46, 87
    - FILLER command 46, 94
  - DATAENCRYPTION
    - configuration file parameter 35
    - keyword
      - BEGIN EXPORT command 70
  - DATE system variables (&SYSDATE and &SYSDATE4) 39, 40
  - DATEFORM command
    - defined 16
    - syntax 74
  - DateTime specifications, table of 89
  - DAY system variable (&SYSDAY) 39, 40
  - dbname specification
    - LOGTABLE command 114, 115
    - with multibyte character sets 45
  - DBQL
    - defined 187
  - defined 181
  - DELETE DATABASE statement, Teradata SQL 18
  - DELETE statement, Teradata SQL 18
  - delimiters 40
    - defined 188
  - DISPLAY command
    - defined 16
    - syntax 75
  - DIT
    - defined 188
  - down AMP 33
  - DROP DATABASE statement, Teradata SQL 18
  - DROP specification, FIELD command 88
  - DWM
    - defined 188
  - DYNAMN entry point
    - for COBOL and PL/I INMOD routines 52
  - dynamn entry point
    - for C INMOD routines 52
    - for SAS/C INMOD routines 52
- E**
- e filename run-time parameter 24
  - EBCDIC character set codes 41
  - ECHO option, ROUTE MESSAGES command 116
  - ELSE command
    - defined 16
    - syntax 96
  - encryption of data
    - run-time option 28
  - END EXPORT command
    - defined 17
    - syntax 77
  - ENDIF command
    - defined 16
    - syntax 96



entry points, for INMOD, OUTMOD, and notify exit routines 52

env\_var environment variable specification  
ACCEPT command 64

ERRLOG configuration file parameter 34, 35

ERRLOG= filename run-time parameter 24

error messages file 35

errors

job script 33

software 33

EUC

defined 189

exclusion join

defined 189

execution time frame

defined 189

EXIT name specification, BEGIN EXPORT command 69

exponential operators, programming considerations 48

EXPORT command

defined 17

syntax 78

expressions, programming considerations 48

## F

failures

defined 189

hardware 33

FastExport

description 13

invocation examples

for UNIX and Windows 135

for z/OS 131

for z/VM 129

involking 23

FASTLOAD format specification

EXPORT command 80

IMPORT command 101

FIELD command

defined 17

syntax 87

field, defined 189

fieldexpr specification, FIELD command 88

fieldname specification

FIELD command 87

FILLER command 94

with multibyte character sets 45

file size

maximum 47

fileid specifications

ACCEPT command 65

DISPLAY command 75

EXPORT command 78, 80

ROUTE MESSAGES command 116

RUN FILE command 118

usage rules for z/OS 65, 76, 82, 103, 117, 119

FILLER command

defined 17

syntax 94

FORMAT specification

EXPORT command 80

IMPORT command 101

## G

GIVE statement, Teradata SQL 18

global rule

defined 190

GRANT statement, Teradata SQL 18

graphic constants, programming considerations 46

GRAPHIC data type specifications

programming considerations 46

GSS

defined 190

## H

hardware

failures 33

hexadecimal form, programming considerations 48

HP-UX

compiling and linking routines 162

## I

IBM-AIX

compiling and linking routines 164

IF command

conditional expression numeric results 96

defined 16

nesting 96

syntax 96

IGNORE charpos specification, RUN FILE command 118

IMPORT command

defined 17

syntax 98

in comments 40

INDICATORS specification, LAYOUT command 105

INFILE fileid specification, IMPORT command 99

infilename standard input file specification 25

INMOD modulename specification, IMPORT command 100

INMOD routines 49

addressing mode on z/VM and z/OS 53

compiling and linking 52, 160

definition 49

entry points 52

examples

using C on Windows 167

using PL/I on z/OS 148

- using SAS C on z/OS 153
- FastExport utility interface 54
- programming languages 50
- programming structure 50
- restart operations--Caution 55
- rules and restrictions 52
- sample programs
  - C language, for UNIX 156
- inner join
  - defined 191
- input text, delimiting 66
- INSERT statement, Teradata SQL 18
- integer specification, EXPORT command 80
- INTEGERDATE specification, DATEFORM command 74
- invocation examples
  - for UNIX and Windows
    - specifying a run file 135
    - specifying error logging 137
    - specifying multiple parameters 138
    - specifying reduced print output 136
    - specifying the character set 136
  - for z/OS
    - specifying error logging 133
    - specifying multiple parameters 134
    - specifying reduced print output 132
    - specifying the character set 132
  - for z/VM
    - specifying error logging 130
    - specifying multiple parameters 131
    - specifying reduced print output 129
    - specifying the character set 129
- invoking FastExport 23

## J

- JIS
  - defined 192
- job scripts
  - definition 59
  - errors 33
  - writing 59
- join, defined 192

## K

- KANJI character set codes 41
- KATAKANA character set codes 41

## L

- LAYOUT command
  - defined 17
  - syntax 104
- layoutname specification
  - IMPORT command 101

- LAYOUT command 104
    - with multibyte character sets 45
- limit specifications, BEGIN EXPORT command 68
- linking routine
  - HP-US 162
  - IBM-AIX 164
  - LINUX 165
  - MP-RAS 161
  - SOLARIS 161
- LINUX
  - compiling and linking routines 165
- LOGDATA command
  - syntax 107
- LOGMECH command
  - syntax 108
- LOGOFF command
  - defined 16
  - syntax 109
  - when permitted 110
- logoff/disconnect messages, description 15
- LOGON command
  - defined 17
  - example 113, 115
  - logon parameters 113
  - syntax 111
  - using with the LOGTABLE command 113, 114
- LOGTABLE command
  - defined 17
  - example 113, 115
  - syntax 114
  - using with the LOGON command 113, 114

## M

- M 'max-sessions command' command specification 27
- maximum
  - file size, programming considerations 47
  - sessions 27
- MAXSESS configuration file parameter 34, 35
- merge join
  - defined 193
- minimum
  - sessions 27
- MINSESS configuration file parameter 34, 35
- minutes specification, BEGIN EXPORT command 68
- MLSCRIPT fileid specification, EXPORT command 80
- MODE specification, EXPORT command 79
- MODIFY DATABASE statement, Teradata SQL 18
- modulename specification
  - EXPORT command 79
  - IMPORT command 100
- MP-RAS
  - compiling and linking routines 161
- MSG stringt specification, BEGIN EXPORT command 69, 70

multibyte character sets, programming considerations 45  
 multiple variables, coding 65

## N

-N 'minsessions command' command specification 27  
 name specification, BEGIN EXPORT command 69  
 name, defined 194  
 Named Pipes Access Module 100  
 nested comments 41  
 nested join  
   defined 194  
 network failure 33  
 nonrecoverable I/O error  
   I/O error 33  
 notify exit routines 49  
   addressing mode on z/VM and z/OS 53  
   compiling and linking 52, 160  
   definition 49  
   entry points 52  
   examples  
     using C on Windows 172  
   programming languages 50  
   programming structure 51  
   rules and restrictions 52  
   sample parameters 158  
 NOTIFY specification, BEGIN EXPORT command 69  
 null, defined 194  
 NULLIF nullexpr specification, FIELD command 88

## O

Object Access filter  
   defined 195  
 OLE DB Access Module 79  
 operators 38  
 OS system variable (&SYSOS) 39  
 oscommand specification, SYSTEM command 122  
 outer join  
   defined 195  
 OUTFILE fileid specification, EXPORT command 78  
 outfile standard output file specification 25  
 OUTLIMIT records specification, EXPORT command 80  
 OUTMOD modulename specification, EXPORT command 79  
 OUTMOD routines 49  
   addressing mode on z/VM and z/OS 53  
   compiling and linking 52  
   definition 49  
   entry points 52  
   examples  
     using C on Windows 170  
     using C on z/VM 142  
     using COBOL on z/VM 140  
   FastExport utility interface 56

input data  
   length pointer 57  
   record pointer 57  
 programming  
   languages 50  
   structure 51  
   restart operations--Caution 57  
   rules and restrictions 52  
 output data length pointer, OUTMOD routine interface 57  
 owner, defined 195

## P

parameter list, FastExport-to-INMOD interface 55  
 parms specification, IMPORT command 101  
 parser, defined 196  
 parsing engine (PE), defined 196  
 password specification  
   LOGON command 111  
   with multibyte character sets 45  
 performance group  
   defined 196  
 PL/I language INMOD routines  
   programming structure 51  
 procedures  
   defined 197  
 product join  
   defined 196  
 product version numbers 3  
 profiles  
   defined 197  
 programming considerations  
   character set specifications 41  
   command conventions 38  
   comments 40  
   date and time variables 40  
   graphic constants 46  
   graphic data types 46  
   multibyte character sets 45  
   restrictions and limitations 47  
 programming structure  
   INMOD routines 50  
   notify exit routines 51  
   OUTMOD routines 51

## Q

queries  
   defined 198  
 query analysis  
   defined 197  
 query management  
   defined 198  
 Query Resource filter  
   defined 198

Query Session utility  
used for FastExport reports 14

## R

-r 'fastexport command' command specification 28  
RC system variable (&SYSRC) 39  
record length specifications, table of 83  
records specification, EXPORT command 80  
reduced print output run-time parameter 23  
RENAME statement, Teradata SQL 19  
REPLACE MACRO statement, Teradata SQL 19  
REPLACE VIEW statement, Teradata SQL 19  
request, defined 199  
reserved words 39  
restart log table 115  
    maintaining 115  
    non-shareability 115  
    specifying 115  
restart operations  
    INMOD routines--Caution 55  
    OUTMOD routines--Caution 57  
restrictions and limitations 47  
    concurrent load tasks 48  
    data file concatenation 48  
    exponential operators 48  
    expressions 48  
    hexadecimal form 48  
    maximum file size 47  
result, defined 199  
results file storage  
    defined 199  
results files  
    defined 199  
results tables  
    defined 199  
retcode specification, LOGOFF command 109  
return codes 31, 48, 110  
REVOKE statement, Teradata SQL 19  
ROUTE MESSAGES command  
    default destinations 117  
    defined 17  
    syntax 116  
row format, examples for z/VM, z/OS, and UNIX 139  
row, defined 199  
RowID join  
    defined 199  
rule  
    defined 199  
RUN FILE command  
    defined 17  
    executing 119  
    nesting 119  
    syntax 118

## S

scheduled requests  
    defined 200  
scripts  
    writing 59  
separator, defined 200  
sequence number, FastExport-to-INMOD interface 55  
session  
    character set  
        AXSMOD 44  
sessions  
    defined 200  
    maximum 27  
    minimum 27  
SESSIONS limit specifications, BEGIN EXPORT command 68  
SET command  
    defined 17  
    syntax 120  
SET SESSION COLLATION statement, Teradata SQL 19  
single sign-on 113  
SLEEP minutes specification, BEGIN EXPORT command 68  
software  
    errors 33  
software releases  
    supported 3  
SOLARIS  
    compiling and linking routines 161  
source file record restrictions 65  
SQL  
    defined 200  
SSO  
    LOGON command  
        single sign-on 113  
startpos specification  
    FIELD command 87  
    FILLER command 94  
statement  
    defined 201  
statement number pointer, OUTMOD routine interface 56  
status codes  
    FastExport-to-INMOD interface 54  
STATUS configuration file parameter 35  
status reporting 14  
stored procedures  
    defined 201  
string specification, BEGIN EXPORT command 69, 70  
supervisory user, defined 201  
syntax  
    how to read 123  
SYSTEM command  
    defined 17  
    syntax 122

**T**

table format, examples for z/VM, z/OS, and UNIX 139  
 table names, with multibyte character sets 45  
 table, defined 201  
 task  
   status reporting 14  
 tdpid specification, LOGON command 112  
 Teradata Database restart 33  
 Teradata SQL statements, supported by FastExport 18  
 termination 31  
   control codes 48  
   return codes 110  
 TEXT format specification  
   EXPORT command 80  
 TEXT string specification, BEGIN EXPORT command 69  
 text string, DISPLAY command 75  
 TIME system variable (&SYSTIME) 39, 40  
 title, defined 202  
 tname specification, LOGTABLE command 114  
 transaction, defined 202  
 type, defined 202

**U**

Unicode  
   character sets 27, 43  
 UNIX  
   C language INMOD example 156  
   C language notify exit parameters 158  
   compiling and linking routines 160  
 UPDATE statement, Teradata SQL 19  
 user 203  
 user groups  
   defined 203  
 USER system variable (&SYSUSER) 40  
 username specification  
   LOGON command 112  
   with multibyte character sets 45  
 USING (parms) specification, IMPORT command 101  
 UTF16  
   character sets 27, 43  
   defined 204  
 UTF8  
   character sets 27, 43  
   defined 203  
 utility variables 76

**V**

var utility variable specifications  
   ACCEPT command 65  
   SET command 120  
 variables  
   declaring, SET command 120

in the IF conditional expression 96  
 number of, ACCEPT command 66  
 substitutions  
   ELSE and ENDIF commands 97  
   SET command 120  
   within comment strings 41  
 utility 76  
 VARTEXT format specification  
   IMPORT command 101  
 version numbers 3

**W**

workgroups  
   defined 204  
 workload limits rule  
   defined 204  
 write operations, conflicting on UNIX systems 75  
 writing job scripts 59

