



Teradata® Tools and Utilities

# **Teradata® Parallel Transporter Reference**

- 17.20

---

Release 17.20

June 2022

# Copyright and Trademarks

Copyright © 1999 - 2023 by Teradata. All Rights Reserved.

All copyrights and trademarks used in Teradata documentation are the property of their respective owners. For more information, see [Trademark Information](#).

## Product Safety

Safety type	Description
 <b>NOTICE</b>	Indicates a situation which, if not avoided, could result in damage to property, such as to equipment or data, but not related to personal injury.
 <b>CAUTION</b>	Indicates a hazardous situation which, if not avoided, could result in minor or moderate personal injury.
 <b>WARNING</b>	Indicates a hazardous situation which, if not avoided, could result in death or serious personal injury.

## Third-Party Materials

Non-Teradata (i.e., third-party) sites, documents or communications ("Third-party Materials") may be accessed or accessible (e.g., linked or posted) in or in connection with a Teradata site, document or communication. Such Third-party Materials are provided for your convenience only and do not imply any endorsement of any third party by Teradata or any endorsement of Teradata by such third party. Teradata is not responsible for the accuracy of any content contained within such Third-party Materials, which are provided on an "AS IS" basis by Teradata. Such third party is solely and directly responsible for its sites, documents and communications and any harm they may cause you or others.

## Warranty Disclaimer

Except as may be provided in a separate written agreement with Teradata or required by applicable laws, all designs, specifications, statements, information, recommendations and content (collectively, "content") available from the Teradata Documentation website or contained in Teradata information products is presented "as is" and without any express or implied warranties, including, but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or noninfringement, which are hereby disclaimed. In no event shall Teradata corporation, its suppliers or partners be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of content, even if advised of the possibility of such damage.

The Content available from the Teradata Documentation website or contained in Teradata information products 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.

The Content available from the Teradata Documentation website or contained in Teradata information products may be changed or updated by Teradata at any time without notice. Teradata may also make changes in the products or services described in the Content at any time without notice.

The Content is subject to change without notice. Users are solely responsible for their application of the Content. The Content does not constitute the technical or other professional advice of Teradata, its suppliers or partners. Users should consult their own technical advisors before implementing any Content. Results may vary depending on factors not tested by Teradata.

## Machine-Assisted Translation

Certain materials on this website have been translated using machine-assisted translation software/tools. Machine-assisted translations of any materials into languages other than English are intended solely as a convenience to the non-English-reading users and are not legally binding. Anybody relying on such information does so at his or her own risk. No automated translation is perfect nor is it intended to replace human translators. Teradata does not make any promises, assurances, or guarantees as to the accuracy of the machine-assisted translations provided. Teradata accepts no responsibility and shall not be liable for any damage or issues that may result from using such translations. Users are reminded to use the English contents.

## Feedback

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

Any comments or materials (collectively referred to as "Feedback") sent to Teradata Corporation will be deemed nonconfidential. Without any payment or other obligation of any kind and without any restriction of any kind, Teradata and its affiliates are hereby free to (1) reproduce, distribute, provide access to, publish, transmit, publicly display, publicly perform, and create derivative works of, the Feedback, (2) use any ideas, concepts, know-how, and techniques contained in such Feedback for any purpose whatsoever, including developing, manufacturing, and marketing products and services incorporating the Feedback, and (3) authorize others to do any or all of the above.

## **Confidential Information**

Confidential Information means any and all confidential knowledge, data or information of Teradata, including, but not limited to, copyrights, patent rights, trade secret rights, trademark rights and all other intellectual property rights of any sort.

The Content available from the Teradata Documentation website or contained in Teradata information products may include Confidential Information and as such, the use of such Content is subject to the non-use and confidentiality obligations and protections of a non-disclosure agreement or other such agreements to protect Confidential Information that you have executed with Teradata.

# Contents

<b>Chapter 1: Teradata PT Utility Commands .....</b>	<b>8</b>
Welcome to Teradata Parallel Transporter Reference .....	8
Overview .....	8
Command Syntax .....	9
<b>Chapter 2: Object Definitions and the APPLY Statement .....</b>	<b>49</b>
Overview .....	49
Object Definition Statements .....	49
Syntax for Attribute Declarations .....	50
DEFINE JOB .....	51
DEFINE SCHEMA .....	56
DEFINE OPERATOR .....	74
APPLY .....	80
<b>Chapter 3: DataConnector Operator .....</b>	<b>91</b>
DataConnector Operator Capabilities .....	91
Syntax for the DataConnector Operator .....	91
Usage Notes .....	124
Operational Considerations .....	136
DataConnector Operator Events .....	138
<b>Chapter 4: DDL Operator .....</b>	<b>141</b>
DDL Operator Capabilities .....	141
Syntax .....	142
Usage Notes .....	151
Operational Considerations .....	154
<b>Chapter 5: Export Operator .....</b>	<b>156</b>
Export Operator Capabilities .....	156
FastExport Utility and the Export Operator .....	156
Syntax .....	158
Usage Notes .....	170
Job Options .....	174
Operational Considerations .....	175
<b>Chapter 6: FastExport OUTMOD Adapter Operator .....</b>	<b>177</b>
FastExport OUTMOD Adapter Operator Capabilities .....	177
Syntax .....	177
Usage Notes .....	181

Job Options .....	182
Operational Considerations .....	182
<b>Chapter 7: FastLoad INMOD Adapter Operator .....</b>	<b>183</b>
FastLoad INMOD Adapter Operator Capabilities .....	183
Syntax .....	183
Usage Notes .....	186
Job Options .....	187
Operational Considerations .....	187
<b>Chapter 8: Load Operator .....</b>	<b>188</b>
Load Operator Capabilities .....	188
FastLoad Utility and the Load Operator .....	189
Syntax .....	190
Usage Notes .....	204
Operational Considerations .....	209
<b>Chapter 9: MultiLoad INMOD Adapter Operator .....</b>	<b>213</b>
MultiLoad INMOD Adapter Operator Capabilities .....	213
Syntax .....	214
Usage Notes .....	217
<b>Chapter 10: ODBC Operator .....</b>	<b>219</b>
ODBC Operator Capabilities .....	219
Syntax .....	220
Usage Notes .....	227
Job Options .....	227
Using Teradata-Provided Branded Drivers .....	233
Operational Considerations .....	235
Restrictions and Limitations .....	237
<b>Chapter 11: OS Command Operator .....</b>	<b>239</b>
OS Command Operator Capabilities .....	239
Syntax .....	239
<b>Chapter 12: Schema Mapping Operator .....</b>	<b>243</b>
Schema Mapping Operator Capabilities .....	243
Syntax .....	243
Attribute Relationships .....	246
Schema Mapping Output Examples .....	248
<b>Chapter 13: SQL Inserter Operator .....</b>	<b>251</b>
SQL Inserter Operator Capabilities .....	251
SQL Inserter Operator and the Load Operator .....	252
Syntax .....	252

Job Options .....	262
Usage Notes .....	262
Operational Considerations .....	262
<b>Chapter 14: SQL Selector Operator .....</b>	<b>265</b>
SQL Selector Operator Capabilities .....	265
Syntax .....	266
Usage Notes .....	277
Operational Considerations .....	279
<b>Chapter 15: Stream Operator .....</b>	<b>280</b>
Stream Operator Capabilities .....	280
TPump Utility and the Stream Operator .....	280
Syntax .....	282
Usage Notes .....	300
Job Options .....	308
Operational Considerations .....	315
<b>Chapter 16: Update Operator .....</b>	<b>318</b>
Update Operator Capabilities .....	318
MultiLoad Utility and the Update Operator .....	319
Syntax .....	321
Usage Notes .....	338
Job Options .....	346
Operational Considerations .....	351
Delete Task Option .....	355
<b>Chapter 17: Notify Exit Routines .....</b>	<b>361</b>
Notify Exit Routines .....	361
DataConnector Operator Events .....	377
Using Notify Exit Routines to Monitor Events .....	379
Compiling and Linking Notify Exit Routines .....	380
<b>Chapter 18: Advanced Database Considerations .....</b>	<b>385</b>
Query Banding Considerations .....	385
Large Objects .....	385
Data Conversions .....	392
Supporting User-Defined-Types and User-Defined-Methods .....	393
Supporting User-Defined-Functions and External-Stored-Procedures .....	395
Connection String .....	397
Identity Token Support .....	398
<b>Chapter 19: Extended Character Sets .....</b>	<b>399</b>
Teradata PT-Supported Character Sets .....	399
Using Extended Character Sets .....	399

Using LONG VARCHAR with Unicode Character Sets . . . . .	404
UTF8/UTF16 Considerations . . . . .	405
<b>Appendix A: How to Read Syntax Diagrams . . . . .</b>	<b>406</b>
<b>Appendix B: Deprecated Syntax . . . . .</b>	<b>411</b>
<b>Appendix C: Restricted Words . . . . .</b>	<b>417</b>
<b>Appendix D: Teradata PT Publications . . . . .</b>	<b>419</b>
<b>Appendix E: Stream Operator Performance Tuning . . . . .</b>	<b>420</b>
<b>Appendix F: Teradata PT Variants on z/OS . . . . .</b>	<b>424</b>
<b>Appendix G: Additional Information . . . . .</b>	<b>426</b>

# Teradata PT Utility Commands

## Welcome to Teradata Parallel Transporter Reference

### Using Teradata Parallel Transporter Reference Content

The *Teradata® Parallel Transporter Reference* content provides reference information about the components of Teradata Parallel Transporter.

### Why Would I Use this Content?

You can use the content in the guide to refer to various command-line utility commands, object definition statements, and learn about different Teradata PT operators.

### How Do I Use this Content?

You can use the content to obtain additional information on the syntax, usage notes, operational considerations, and other relevant information on the following components:

- Command-line utility commands
- Object definition statements
- Operators

### How Do I Get Started?

Read the following topics before you use the commands, statements, or operators in the document:

- Read the [Overview](#) section.
  - Read the [Release Compatibility](#) section.
  - Read the [Syntax Diagram Conventions](#) section.
  - Read the [Restricted Words](#) section.
  - Read the [Teradata PT-Supported Character Sets](#) section.

### References to Other Relevant Content

See [Teradata PT Publications](#) and [Additional Information](#).

## Overview

Starting in Teradata PT 17.00:

- Teradata PT (script-based and Easy Loader) is only 64-bit for the following operating systems:
  - IBM AIX

- Linux
- macOS
- Solaris
- Windows
- Teradata PT is only 32-bit for the z/OS operating system.
- On z/OS, there are two variants of TPT:
  - TDP
  - Gateway

For more information, see [Teradata PT Variants on z/OS](#).

The following topics describe Teradata PT command line utility commands:

This command...	Does the following:
<a href="#">tbuild</a>	Defines and executes a Teradata PT job.
<a href="#">tdload</a>	Loads data from a delimited file into a database table without requiring a job script.
<a href="#">tdlog</a>	Extracts the logs produced m running a Teradata PT job.
<a href="#">tlogview</a>	Displays the contents of the log files produced from running a Teradata PT job.
<a href="#">twbcmd</a>	Allows the modification of an active Teradata PT job and the retrieval of job status.
<a href="#">twbertbl</a>	Extracts the error information from the acquisition phase error table.
<a href="#">twbkill</a>	Terminates all Teradata PT tasks within an application.
<a href="#">twbrmcp</a>	Removes all checkpoint files for a specified user ID or job name.
<a href="#">twbstat</a>	Displays Teradata PT application status. <b>Note:</b> On z/OS, Teradata PT does not support the invocation of the Teradata PT command line utility commands in the USS (UNIX System Services) environment.

## Command Syntax

The following sections provide the syntax and a brief description of the options for each Teradata PT utility command.

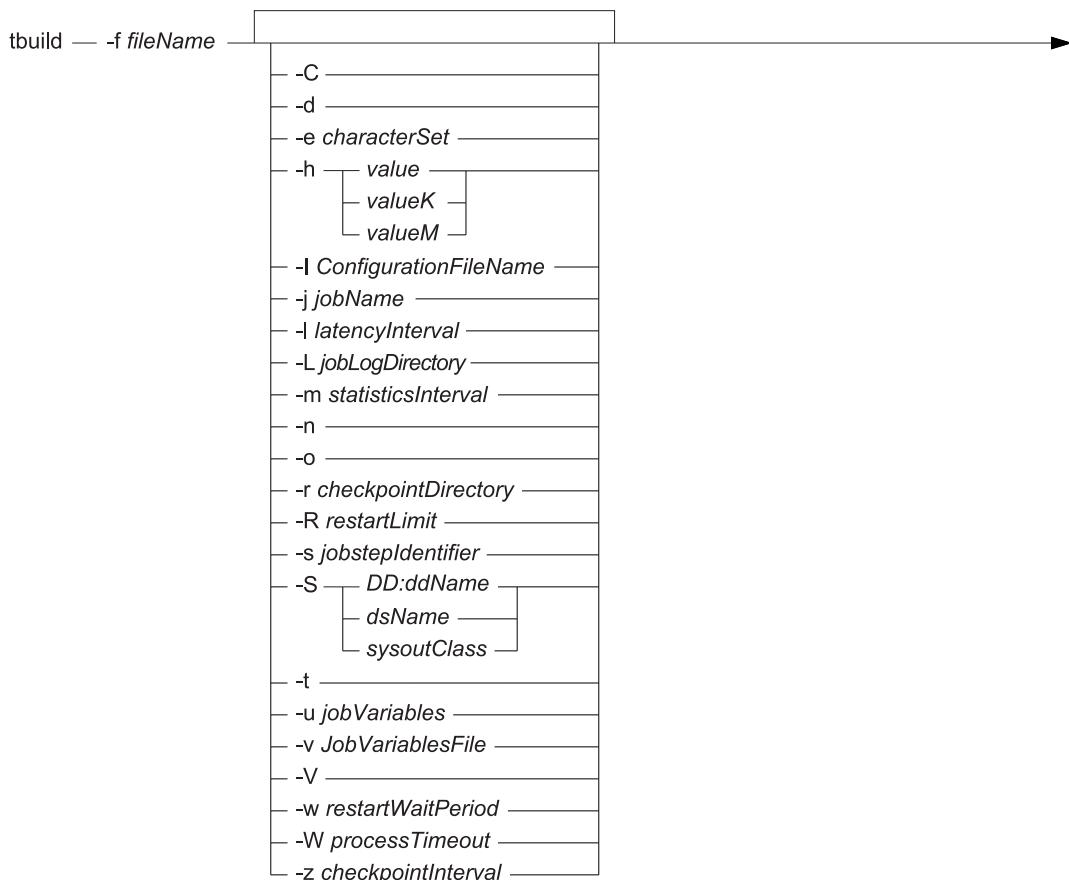
## tbuild

### Purpose

The **tbuild** command defines and executes Teradata PT job scripts. The job scripts must conform to Teradata PT syntax rules.

### Syntax

The tbuild command takes as its primary argument the file containing the Teradata PT job script. You can specify tbuild options in any order.



where:

Syntax Element	Description
-C	<p>Provides a more even distribution of data to consumer operators by instructing producer operators and their underlying data streams to ship data blocks to target consumer operators in a cyclical or round-robin manner.</p> <p>By default, Teradata PT data streams send a block of data to the first ready /available consumer operator found in the producer's channel set array, only switching processes when the first consumer operator is too busy to accept more input. Using the -C option ensures that data are evenly distributed among consumer operators.</p>
-d	<p>Enables debug trace functions for all tasks.</p> <p>Using this option outputs trace messages and return codes of all internal functions invoked on behalf of <b>tbuild</b>. Internal error condition codes and trace messages are usually helpful for debugging issues occurring at the infrastructure level. However, when using this option along with the trace options provided by the operators, a full trace of the job can be obtained. If this option is not specified, the debug trace function is disabled.</p>
-e <i>characterSet</i>	<p>Specifies the character set encoding of the script. This option also ensures that files in big endian and little endian format are processed correctly regardless of the platform's native encoding.</p> <p>If not specified, the default character set is 7-bit ASCII compatible on UNIX OS, Windows and EBCDIC on z/OS.</p> <p>-e <i>characterSet</i> is required if the job script is encoded in UTF-16.</p> <p>All values are case insensitive and may be used with or without hyphens, that is, UTF-16, UTF16, utf16-be, and any similar notations are all valid. The following values are valid for <i>characterSet</i>:</p> <ul style="list-style-type: none"> <li>• UTF-16 (Preferred notation) Endianess defaults to that of the client platform</li> <li>• UTF-8 (Preferred notation) Specifying UTF-8 is optional. Specifying UTF-8 for a non-UTF-8 script generates an error. Endianess is irrelevant for UTF-8 encoding.</li> <li>• UTF-16LE (Preferred notation) Specifies a job script with little-endian encoding. If executed on a big endian platform, script encoding is converted to big-endian encoding before being executed. Not permitted as a client session character set name.</li> <li>• UTF-16BE (Preferred notation) Specifies a job script with big-endian encoding. If executed on a little-endian platform, script encoding is converted to little-endian encoding before being executed. Not permitted as a client character set name.</li> </ul>
-f <i>fileName</i>	Specifies the Teradata PT job script file.
-h <i>value</i> -h <i>valueK</i> -h <i>valueM</i>	<p>Specifies the size of the shared memory used among the processes of a Teradata PT job.</p> <p>Options include:</p> <ul style="list-style-type: none"> <li>• -h <i>value</i>, where <i>value</i> specifies the shared memory size in bytes and can range from 1,048,576 bytes to 4,294,967,295 bytes.</li> </ul>

Syntax Element	Description
	<ul style="list-style-type: none"> <li>-h <i>valueK</i>, where <i>valueK</i> specifies the shared memory size in kilobytes and can range from 1024 K (1,048,576 bytes) to 4,194,304K (4,294,967,295 bytes).</li> <li>-h <i>valueM</i>, where <i>valueM</i> specifies the shared memory size in megabytes and can range from 1M (1,048,576 bytes) to 4096M (4,294,967,295 bytes).</li> </ul> <p>For example,</p> <pre>tbuild -h 24M</pre> <p>If the -h option is not specified, Teradata PT allocates shared memory according to the calculations outlined in <i>Teradata® Parallel Transporter User Guide</i>, B035-2445.</p> <p>If the -h option is invalid, a warning message is issued, and allocates shared memory according to the calculations outlined in <i>Teradata® Parallel Transporter User Guide</i>, B035-2445.</p> <p>The -h memory size is invalid, and will be ignored.</p> <p>If the value of the -h option falls outside of the allowed range, a warning is issued and the size is adjusted to the appropriate minimum or maximum value before allocation. The following messages are issued if the -h option is out of the allowed range:</p> <ul style="list-style-type: none"> <li>The -h memory size exceeds the maximum allowed value, and will be lowered to 4294967295.</li> <li>The -h memory size is less than the minimum allowed value, and will be raised to 1048576.</li> </ul>
-I <i>ConfigurationFileName</i>	<p>Specifies a desired configuration file when it's not possible to access the following files:</p> <ul style="list-style-type: none"> <li>The global configuration file twbcfg.ini, located under TPT install directory</li> <li>The local configuration file .twbcfg.ini, located under the user's home directory</li> </ul> <p>The following global parameters can be defined in a user-specified configuration file:</p> <ul style="list-style-type: none"> <li><i>GlobalAttributeFile</i></li> <li><i>CheckpointDirectory</i></li> <li><i>LogDirectory</i></li> </ul> <p>These parameter definitions must be specified using the following syntax:</p> <pre>&lt;parameter&gt; = &lt;single-quoted string&gt;</pre> <p>For example, on a Unix system:</p> <pre>CheckpointDirectory='/opt/teradata/client/16.20/tbuild/checkpoint' LogDirectory='/opt/teradata/client/16.20/tbuild/logs'</pre> <p>The -I option is only supported on Windows and Unix platforms.</p>
-j <i>jobName</i>	Optional job name, but strongly recommended so each job can have a unique checkpoint file. The default, if you do not enter a <i>jobName</i> , is the user name followed by a hyphen (“-”) and a generated Teradata PT job sequence number as follows:

Syntax Element	Description
	<p>&lt;user name&gt;-&lt;job sequence number&gt;</p> <p><b>Note:</b> If multiple jobs are to be simultaneously run under the same logon session, specify a unique name for each job because each job must write its own checkpoint file. Without a unique name, concurrent jobs try to write to the same checkpoint file. This causes the jobs to fail.</p>
-l <i>latencyInterval</i>	<p>Optional latency interval, specified in seconds, for flushing stale buffers. Latency interval is used exclusively with the Stream operator.</p> <p>If no value is specified, data is read from the data stream until its buffer is full. Then all buffered records are written to the database.</p>
-L <i>jobLogDirectory</i>	<p>Designates the location of the Teradata PT files created during job execution. <i>jobLogDirectory</i> is the full path name of the directory in which the Teradata PT files are stored.</p> <p>This option is not supported on z/OS.</p>
-m <i>statisticsInterval</i>	<p>Specifies a time interval, in seconds, in which to collect statistical information about operators.</p> <p>If this option is not specified, no statistical information is collected.</p>
-n	<p>Specifies that the job can continue to run even if a job step returns an error.</p> <ul style="list-style-type: none"> <li>If a job step receives a non-zero exit code, the job will continue.</li> <li>If a step exits with a status of “failure” (exit code higher than 4), the subsequent steps will bypass the checkpoint file left by the previously failed step.</li> </ul> <p>The required syntax is:</p> <pre>tbuild -f &lt;scriptFile&gt; -n</pre> <p>If this option is <i>not</i> specified, the job will stop if a step fails.</p>
-o	<p>Writes the consumer private logs to standard output after a job completes.</p> <p>If the -o option is not specified, logs will not be written to standard output.</p>
-r <i>checkpointDirectory</i>	<p>Specifies that checkpoint files are to be stored in a directory called “CheckpointDirectory.”</p> <p>If the -r option is not specified, then checkpoint files will be stored in the default checkpoint directory that is defined in the Teradata PT configuration file.</p>
-R <i>restartLimit</i>	<p>Option that overrides the default value of five tries at automatic (job) restart. If you specify -R, enter a value or the system will reject the command and return an error. The restartLimit value must be between 0 and 512. The value zero prevents automatic job restart.</p>
-s <i>jobstepIdentifier</i>	<p>Directs job execution to start at the specified job step.</p> <p>Job steps are identified by a job step name in a script or by an implicit job step number that corresponds to the physical order of job steps in a script, such as 1, 2, 3.</p>

Syntax Element	Description
	<p>The job will start at the specified job step, skipping over all job steps that come before it in the job script.</p> <p>This option also removes checkpoint files:</p> <ul style="list-style-type: none"> <li>• used by any skipped steps</li> <li>• used by a previous execution of the same job</li> <li>• that would have been carried forward because of a failure or premature termination of one or more of the preceding steps</li> </ul> <p>Since checkpoint files subject to these conditions are not valid for subsequent steps, the system will remove them before starting the step specified for the -s option.</p> <p><b>Note:</b></p> <p>The -s option formerly supported silent mode. Silent mode is no longer available.</p>
-S <i>DD:ddname</i> -S <i>dsName</i> -S <i>sysoutClass</i>	<p>Enables <b>tbuild</b> to write both the private and public logs to a specified location for z/OS platforms. Choose from the following logDestination options:</p> <ul style="list-style-type: none"> <li>• To specify a ddname: <b>tbuild -S logDestination</b> where logDestination is <i>DD:&lt;ddname&gt;</i>, the target ddname for the log file.</li> <li>• To specify a dsname: <b>tbuild -S logDestination</b> where logDestination is <i>&lt;dsname&gt;</i>, the target dataset name for the logfile.</li> </ul> <p><b>Note:</b></p> <p>A fully qualified dataset name is indicated by enclosing the name in single quote marks.</p> <ul style="list-style-type: none"> <li>• To specify a SYSOUT class: <b>tbuild -S logDestination</b> where logDestination is <i>&lt;sysout class&gt;</i>, the SYSOUT class for the log file.</li> </ul> <p>If the -S option is not specified, <b>tbuild</b> does not write the private or public log to a specified location. However, <b>tbuild</b> still writes the logs to the dataset referenced by the JOBLOG DD statement.</p>
-t	Enables the trace option for all tasks. If this option is not specified, trace is disabled.
-u <i>jobVariables</i>	Allows you to specify job variable values on the command line. Use this option to specify job variable assignments on the command line for the current execution of the job script.

Syntax Element	Description
	<p><b>Note:</b></p> <p>There are multiple sources of job variable assignments within a job script, which are invoked in a priority order hierarchy. For more information, see “Using Job Variables” and related topics in the <i>Teradata® Parallel Transporter User Guide</i>, B035-2445.</p> <p>In the following example, UsrID and Pwd are defined as job variables and they are used to supply runtime values for the UserName and UserPassword attributes:</p> <pre>ATTRIBUTES (     VARCHAR UserName = @UsrID,     VARCHAR UserPassword = @Pwd );</pre> <p>The following command supplies runtime values for these job variables:</p> <pre>tbuild -f scriptFileName -u "UsrID = 'John Doe', Pwd = 'ABC123'"</pre>
<b>-v jobVariablesFile</b>	<p>Allows job variable values to be specified in an external file. This is similar to the <b>-u</b> option for assigning values at run time to job variables specified in a script, except that the variable values are stored in an external file that is referenced by <b>tbuild</b> through the following command:</p> <pre>tbuild -f scriptFileName -v jobVariablesFile</pre> <p>Where <i>jobVariableFile</i> is a file which contains the values for substitution:</p> <pre>UsrID = 'John Doe', Pwd = 'ABC123'</pre> <p>If you are not executing the <b>tbuild</b> command in the directory where the job variables file is stored, <i>jobVariablesFile</i> must be a fully qualified filename. The job variables file may be saved in ASCII, UTF-8, or UTF-16, both with and without a UTF byte order mark.</p>
<b>-V</b>	<p>Displays the Teradata PT version number without running a job. Do not use with any other option. The option works only on UNIX and Windows platforms.</p>
<b>-w restartWaitPeriod</b>	<p>Specifies a time interval, in seconds, between two restarts. If this option is not specified, there will be no wait period between auto restarts. The valid restart wait period value must be a positive integer between 1 and 86400 (seconds).</p>
<b>-W processTimeout</b>	<p>Specifies the number of seconds TPT will wait for a subprocess to spawn. The value can be an integer between 1 and 900. Consider using this option if jobs frequently fail with the following error: Error: Timed out waiting for task initialization.</p>

Syntax Element	Description
	If the -W option is not specified, TPT will use the default of 120 seconds. If the -W option is invalid (that is, 0 or non-numeric), TPT will use the default value of 120 seconds. The following message is issued if the -W option is invalid: <code>TPT_INFRA: TPT04215: Warning: The -W timeout value is invalid and will be ignored.</code> If the -W option is greater than the maximum allowed value of 900 seconds, TPT will use the maximum value of 900 seconds and display the following message: <code>TPT_INFRA: TPT04214: Warning: The -W timeout value is greater than the maximum allowed value and will be lowered to 900.</code>
<code>-z checkpointInterval</code>	Option that specifies a time interval, in seconds, between checkpoints. If this option is not specified, there will be no interval checkpointing, unless a checkpoint interval is specified in the job script. If a checkpoint interval is specified in both places, the -z option specification takes precedence. The valid checkpoint interval values are between 0 and 86400 (seconds).

## Use of the Redirection Symbol

On the Unix/Linux platforms, the use of the redirection symbol with options, such as:

```
<%-
```

```
<&-
```

will not work on the tbuild command line.

## tdload

### Purpose

The Teradata PT Easy Loader command, `tdload`, loads data into a database table from a comma-delimited flat file or another database table without requiring a Teradata PT job script to be written.

---

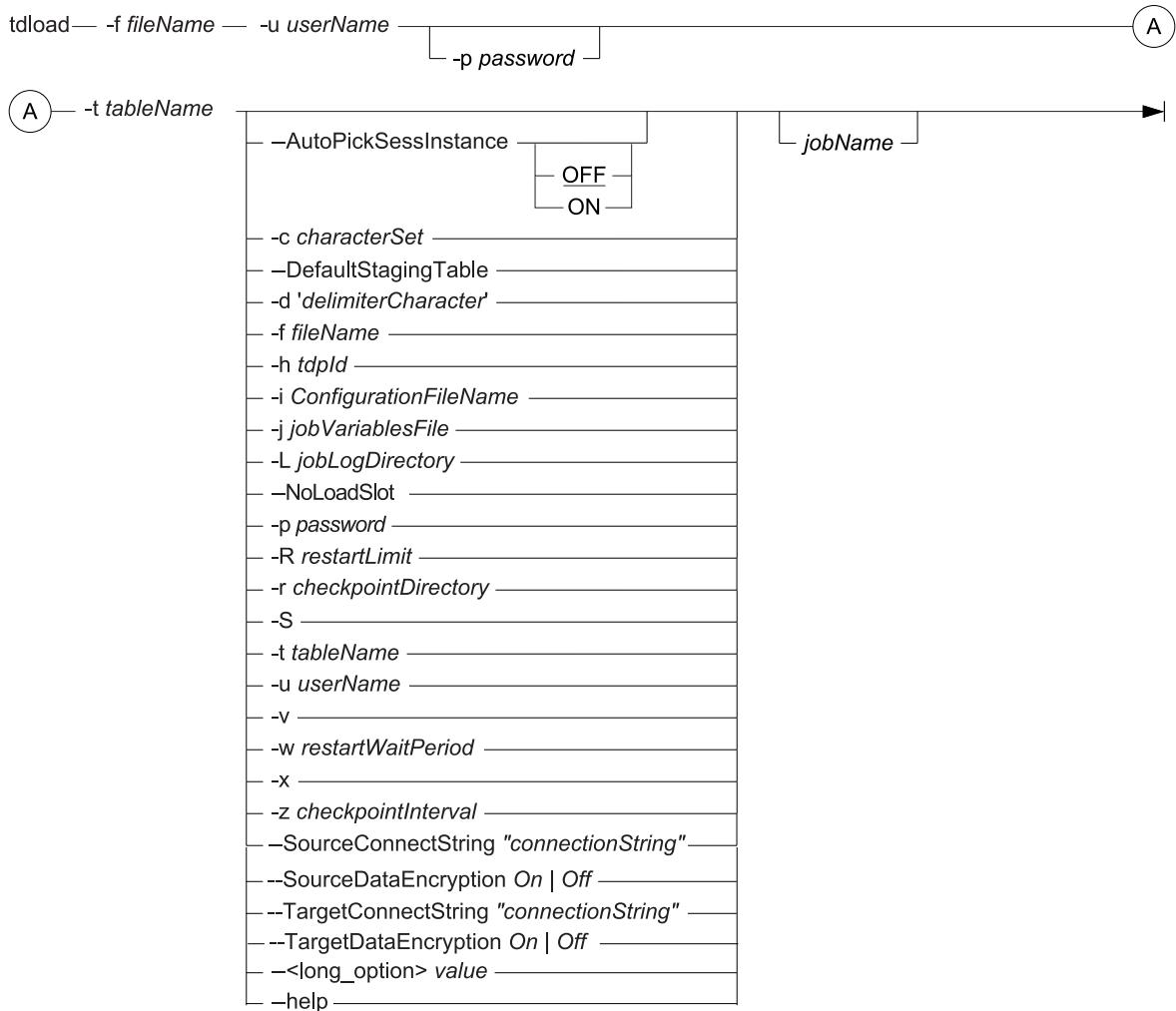
#### Note:

The command is supported on all platforms, except z/OS.

---

## Syntax

`tdload` has the following syntax:



where *jobOptions* are:

Syntax Element	Description
AutoPickSessInstance ON/OFF	Tells TPT Easy Loader to pick the session/instance count automatically. The default value of this option is OFF.  For more details, refer to "Picking Sessions and Instances Automatically in Teradata PT Easy Loader" in <i>Teradata® Parallel Transporter User Guide</i> , B035-2445.
-c characterSet	The character set encoding of the flat file. This option sets the client session character set.  Specify this option if the flat file is not an ASCII file.

Syntax Element	Description
<code>-d "delimiterCharacter"</code>	<p>The delimiter character used to separate the fields of the data records in the delimited format flat file.</p> <p><b>Note:</b></p> <p>The default delimiter character is a comma (","). This is different from the usage in Teradata PT scripts where the default delimiter character is the pipe character (" "). You must specify this option if the delimiter character in your flat file is not a comma.</p>
<code>-f fileName</code>	<p>Required. The name of the flat file containing the data to be loaded. If you are not executing the <code>tdload</code> command in the directory where the flat file is stored, <code>fileName</code> must be a fully qualified flat file name.</p> <ul style="list-style-type: none"> <li>The <code>-f</code> (or <code>--SourceFileName</code>) option and the <code>--SourceTable</code> option are mutually exclusive. An error is returned if both options are specified on the <code>tdload</code> command line.</li> <li>The <code>-f</code> (or <code>--SourceFileName</code>) option and the <code>--SelectStmt</code> option are mutually exclusive. An error is returned if both options are specified on the <code>tdload</code> command line.</li> </ul>
<code>-h tdpId</code>	<p>The name by which the Teradata Data Warehouse Appliance 2xxx is known to the network. If the option is not specified, the default host name will be used. Recommendation: Specify this option to make sure you are connecting to the correct system.</p>
<code>-l ConfigurationFileName</code>	<p>Specifies a desired configuration file when it's not possible to access the following files:</p> <ul style="list-style-type: none"> <li>The global configuration file <code>twbcfg.ini</code>, located under TPT install directory</li> <li>The local configuration file <code>.twbcfg.ini</code>, located under the user's home directory</li> </ul> <p>The following global parameters can be defined in a user-specified configuration file:</p> <ul style="list-style-type: none"> <li><code>GlobalAttributeFile</code></li> <li><code>CheckpointDirectory</code></li> <li><code>LogDirectory</code></li> </ul> <p>These parameter definitions must be specified using the following syntax:  <code>&lt;parameter&gt; = &lt;single-quoted string&gt;</code></p> <p>For example, on a Unix system:</p> <pre>CheckpointDirectory='/opt/teradata/client/16.20/tbuild /checkpoint' LogDirectory='/opt/teradata/client/16.20/tbuild/logs'</pre> <p>The <code>-l</code> option is only supported on Windows and Unix platforms.</p>
<code>-j jobVariablesFile</code>	<p>The name of the job variables file. If you are not executing the <code>tdload</code> command in the directory where the job variables file is stored, <code>jobVariablesFile</code> must be a fully qualified filename.</p>

Syntax Element	Description
	The job variables file may be saved in ASCII, UTF-8, or UTF-16, both with and without a UTF byte order mark.
<b>-L <i>jobLogDirectory</i></b>	Option that designates the location of the Teradata PT files created during job execution. <i>jobLogDirectory</i> is the full path name of the directory in which the Teradata PT files are stored. This option is not supported on z/OS.
<b>-p <i>password</i></b>	The password of the specified user. If the option is not specified, <i>tdload</i> prompts you for a password.
<b>-R <i>restartLimit</i></b>	Option that overrides the default value of five tries at automatic (job) restart. If you specify -R, enter a value or the system will reject the command and return an error. The <i>restartLimit</i> value can be any whole number greater than zero or equal to zero. The value zero prevents automatic job restart.
<b>-r <i>checkpointDirectory</i></b>	Option that specifies that checkpoint files are to be stored in a directory called "CheckpointDirectory." If the -r option is not specified, then checkpoint files will be stored in the default checkpoint directory that is defined in the Teradata PT configuration file.
<b>-S</b>	Saves the Teradata PT script generated by the command.
<b>-t <i>tableName</i></b>	Required. The name of the target table. <b>Note:</b> If the target table resides in a database that is different from the default database of the specified user, you must also use the --TargetWorkingDatabase option.
<b>-u <i>userName</i></b>	Required. The logon id of the user with access privileges to the target table.
<b>-v</b>	Option that displays the version number of <i>tdload</i> without running a job. Do not use with any other option. The option works only on UNIX and Windows platforms.
<b>-w <i>restartWaitPeriod</i></b>	Specifies a time interval, in seconds, between two restarts. If this option is not specified, there will be no wait period between auto restarts. The valid restart wait period value must be a positive integer between 1 and 86400 (seconds).
<b>-x</b>	Enables debugging.
<b>-z <i>checkpointInterval</i></b>	Option that specifies a time interval in seconds between checkpoints. If this option is not specified, there will be no interval checkpointing. The valid checkpoint interval values are between 0 and 86400 (seconds).
<b>--help</b>	Displays help.

Syntax Element	Description
--DefaultStagingTable	<p>The --DefaultStagingTable option tells TPT Easy Loader to create a NOPI staging table and load data to the staging table instead of the target table. The staging table will be created on the target database.</p> <p>The name of the staging table will be:  <code>&lt;target table&gt;_STG</code></p> <p>When data is coming from a file, TPT Easy Loader obtains the table definition from the target table, convert the non-VARCHAR columns to equivalent VARCHAR columns, and create the staging table which comprises all VARCHAR columns.</p> <p>This will allow data that might otherwise cause issues and/or errors, like data conversion errors, to be loaded into the VARCHAR columns in the staging table.</p> <p>After the data is loaded into the staging table, the user can create the necessary INSERT-SELECT statement to move the data from the staging table to the target table.</p> <p>When data is from a source table, TPT Easy Loader obtains the table definition from the source table, convert the non-VARCHAR columns to equivalent VARCHAR columns, and create the staging table which comprises all VARCHAR columns.</p> <p>When data is from a SELECT request of a source table, TPT Easy Loader obtains the schema of the SELECT request, convert the non-VARCHAR columns to equivalent VARCHAR columns, and create the staging table which comprises all VARCHAR columns.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>TPT Easy Loader will use Load operator to load to the staging table, because Load operator uses the fastest protocol (FastLoad) to load data into an empty table.</li> <li>If the total length of the target table and the "_STG" suffix exceeds the database maximum size for a table name, TPT Easy Loader will truncate the target table name and add the "_STG" suffix, so that the resulting length of the staging table does not exceed the database maximum size for a table name.</li> <li>The staging table must not exist. If the staging table already exists, TPT Easy Loader will terminate the job with an error message.</li> <li>The user can directly load to the target table without loading to the staging table by not specifying the --DefaultStagingTable option.</li> <li>The --StagingTable and --DefaultStagingTable options are mutually exclusive. TPT Easy Loader will terminate the job with an error message if both options are specified.</li> <li>The --DefaultStagingTable option cannot be specified in the job variable file. It can only be specified on the command line. A job variable file requires a value and the --DefaultStagingTable option does not have a value.</li> </ul>
--NoLoadSlot	<p>This option tells TPT Easy Loader not to use a load slot. It implies this on both source and target machines if there are 2 of them; otherwise, just the source or just the target.</p> <p>This option can be overridden by specifying "Load/Update/Export/Stream" prefixes to any of the job variables.</p>

Syntax Element	Description
	<p><b>Note:</b></p> <p>The --NoLoadSlot, --StagingTable, and --DefaultStagingTable options are mutually exclusive. TPT Easy Loader will terminate the job with the following error message if these options are specified:</p> <pre>TPT_INFRA: TPT05565: Error: Option --NoLoadSlot is mutually exclusive with --StagingTable and -- DefaultStagingTable.</pre> <p><b>Note:</b></p> <p>The --NoLoadSlot option cannot be specified in the job variable file. It can only be specified on the command line. A job variable file requires a value and the --NoLoadSlot option does not have a value.</p>
<long_option> value	<p>Easy Loader supports all standard job variables which are defined within the operator templates as well as generic job variables. These generic job variables are derived from operator templates and allow Easy Loader to automatically make operator decisions. To construct a generic job variable, simply remove the operator name from the template job variable, and add 'Source' or 'Target' depending on if the operator is a producer or consumer, respectively.</p> <p>For example, the trace level job variable for the Load operator is "LoadTraceLevel", to make this generic, specify it as "TargetTraceLevel". If Load is the consumer operator chosen by Easy Loader, "TargetTraceLevel" will be automatically mapped to the load operator.</p> <p>See the Teradata PT Easy Loader section in <i>Teradata® Parallel Transporter User Guide</i>, B035-2445 for more examples and instructions.</p>

Where commonly used long options are:

Long Option	Description
--InsertStmt "INSERT statement;"	<p>The Insert statement to be used for the loader operator.</p> <pre>tdload --InsertStmt "INSERT &lt;table&gt; (:COL1, :COL2);"</pre> <p>This option allows for the data coming from the producer to be sent selectively and in a different order than how it is defined in the source schema. Otherwise, the \$INSERT macro is used, and the schema of the source table must then need to be compatible with the schema of the target table.</p>
--SelectStmt "SELECT statements;"	<p>SELECT statement performs data selection from database tables. Since the SELECT statement can have space/blank characters, the entire statement should be enclosed in double quotation marks ("") when specified on the command line, as follows:</p> <pre>tdload --SelectStmt "sel * from src_tbl;"</pre> <p>SELECT statement requests cannot:</p> <ul style="list-style-type: none"> <li>• Specify a USING modifier.</li> <li>• Access non-data tables, such as SELECT DATE or SELECT USER.</li> </ul>

Long Option	Description
	<ul style="list-style-type: none"> <li>• Be satisfied by one or two AMPs, such as a SELECT statement that accesses rows based on the primary index or unique secondary index of a table.</li> <li>• Contain BLOB (Binary Large Object) or CLOB (Character Large Object) data types.</li> <li>• Contain JSON (JavaScript Object Notation) data type.</li> <li>• Contain XML data type.</li> </ul> <p>The --SelectStmt option and the -f (or --SourceFileName) option are mutually exclusive. An error is returned if both are defined on the <b>tdload</b> command line.</p> <p>If both the --SelectStmt option and the --SourceTable option are defined, the --SourceTable option is ignored and a warning message is returned to the console.</p>
--SourceAccountId <i>accountId</i>	The account associated with the specified user.
--SourceConnectionString "connectionString"	<p>Specifies the connection string when extracting data from tables. The connection string will be passed to CLI. CLI will validate the connection string.</p> <p>For information on connection string, see <i>Teradata® Call-Level Interface Version 2 Reference for Workstation-Attached Systems</i>, B035-2418.</p>
--SourceDataEncryption <i>On</i>   <i>Off</i>	<p>Enables full encryption of SQL requests, responses, and data when extracting data from tables.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>• On = All SQL requests, responses, and data are encrypted.</li> <li>• Off = No encryption occurs (default).</li> </ul> <p><b>Note:</b> This option does not include logon encryption.</p>
--SourceFileName <i>fileName</i>	<p>Required. The name of the flat file that contains the data to be loaded.</p> <ul style="list-style-type: none"> <li>• The --SourceFileName (or -f) option and the --SourceTable option are mutually exclusive. An error is returned if both options are specified on the <b>tdload</b> command line.</li> <li>• The --SourceFileName (or -f) option and the --SelectStmt option are mutually exclusive. An error is returned if both options are specified on the <b>tdload</b> command line.</li> </ul>
--SourceInstances <i>number</i>	The number of instances used to extract data from the database. The default value is 1.
--SourceMaxSessions <i>number</i>	The maximum sessions to be used in extracting data from tables. The default value is 32.
--SourceMinSessions <i>number</i>	The minimum sessions to be used in extracting data from tables. The default value is 1.

Long Option	Description
--SourceTable <i>tableName</i>	<p>The name of the source table.</p> <p>Note: If the source table resides in a database that is different from the default database of the specified user, you must also use the --SourceWorkingDatabase option.</p> <p>The option --SourceTable and the -f (or --SourceFileName) option are mutually exclusive. An error is returned if both of them are specified on the <b>tdload</b> command line.</p> <p>If both the --SourceTable option and the --SelectStmt option are defined, the option --SourceTable is ignored and a warning message is returned to the console.</p>
--SourceTextDelimiter " <i>delimiterCharacter</i> "	The delimited character used to separate the fields of the data records in the delimited format flat file.
--SourceTpId <i>tpId</i>	The name by which the database is known to the network.
--SourceTraceLevel <i>value</i>	<p>Enables trace messages in the operator that reads data from flat files.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>• 'Yes' = <b>tdload</b> sets the value 'all' to the attribute 'TraceLevel' of the appropriate operator.</li> <li>• 'No' = TraceLevel is turned off (default).</li> </ul>
--SourceUserName <i>userName</i>	The logon id of the user with access privileges to the source table.
--SourceUserPassword <i>password</i>	<p>The password of the specified user.</p> <p>If the option is not specified, <b>tdload</b> will prompt you for a password.</p>
--SourceWorkingDatabase <i>databaseName</i>	The database where the source table is located.
--StagingTable <i>stagingTableName</i>	<p>The --StagingTable option tells TPT Easy Loader to create a NOPI staging table with the given &lt;staging table name&gt; and load data to the staging table instead of the target table.</p> <p>The staging table will be created on the target database.</p> <p>When data is coming from a file, TPT Easy Loader obtains the table definition from the target table, convert the non-VARCHAR columns to equivalent VARCHAR columns, and create the staging table which comprises all VARCHAR columns.</p> <p>This will allow data that might otherwise cause issues or errors, like data conversion errors, to be loaded into the VARCHAR columns in the staging table.</p> <p>After the data is loaded into the staging table, the user can create the necessary INSERT-SELECT statement to move the data from the staging table to the target table.</p> <p>When data is from a source table, TPT Easy Loader obtains the table definition from the source table, converts the non-VARCHAR columns to equivalent VARCHAR columns, and creates the staging table which will comprise all VARCHAR columns.</p> <p>When data is from a SELECT request of a source table, TPT Easy Loader will obtain the schema of the SELECT request, convert the</p>

Long Option	Description
	<p>non-VARCHAR columns to equivalent VARCHAR columns, and create the staging table which will comprise all VARCHAR columns.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>• TPT Easy Loader will use Load operator to load to the staging table, because Load operator uses the fastest protocol (FastLoad) to load data into an empty table.</li> <li>• The staging table must not exist. If the staging table already exists, TPT Easy Loader will terminate the job with an error message.</li> <li>• The staging table can be qualified with a database name.</li> <li>• The user can directly load to the target table without loading to the staging table by not specifying the --StagingTable option.</li> <li>• The --StagingTable and --DefaultStagingTable options are mutually exclusive. TPT Easy Loader will terminate the job with an error message if both options are specified.</li> <li>• The --StagingTable option can be specified in the job variable file as follows:  <code>StagingTable = 'tablename'</code>            If you specify a value for the --StagingTable command line option and a value for --StagingTable option in the job variable file, the value on the command line option takes precedence.</li> </ul>
--SharedMemorySize	<p>The --SharedMemorySize option allows user to specify a desired amount of shared memory to be allocated for a running job. The value provided is then used in the option (-h) of the tbuild command.</p> <p><b>Note:</b></p> <p>Teradata PT Easy Loader does not perform any error checking for the SharedMemorySize value; the value provided to tdload command is passed directly to tbuild command as is.</p>
--TargetAccountId <i>accountId</i>	<p>The account associated with the specified user.</p>
--TargetConnectionString " <i>connectionString</i> "	<p>Specifies the connection string in the load job. The connection string will be passed to CLI. CLI will validate the connection string.</p> <p>For information on connection string, see <i>Teradata® Call-Level Interface Version 2 Reference for Workstation-Attached Systems</i>, B035-2418.</p>
--SourceDataEncryption On   Off	<p>Enables full encryption of SQL requests, responses, and data when extracting data from tables.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>• On = All SQL requests, responses, and data are encrypted.</li> <li>• Off = No encryption occurs (default).</li> </ul> <p><b>Note:</b></p> <p>This option does not include logon encryption.</p>
--TargetErrorLimit <i>number</i>	<p>The maximum errors allowed in data records.</p> <p>When the number of errors encountered exceeds this number, the load job terminates.</p>

Long Option	Description
	The default value is 1.
--TargetMaxSessions <i>number</i>	The maximum sessions to be used in the load job. The default value is 32.
--TargetMinSessions <i>number</i>	The minimum sessions to be used in the load job. The default value is 1.
--TargetTable <i>tableName</i>	Required. The name of the target table.
--TargetTdpld <i>tdpld</i>	The name by which the database is known to the network.
--TargetTraceLevel <i>value</i>	Enables trace messages in the operator that loads data to the target table. Valid values are: <ul style="list-style-type: none"><li>• 'Yes' = tload sets the value 'all' to the attribute TraceLevel of the appropriate operator.</li><li>• 'No' = TraceLevel is turned off (default).</li></ul>
--TargetUserName <i>userName</i>	Required. The logon id of the user with access privileges to the target table.
--TargetUserPassword <i>password</i>	The password of the specified user.
--TargetWorkingDatabase <i>databaseName</i>	The database containing the target table. <b>Note:</b> This option is required if the target table resides in a database that is different from the default database of the specified user.

**Note:**

“--< long\_option> *value*” is any of the generic or operator template job variables. For more information on usage, see the Teradata PT Easy Loader section of *Teradata® Parallel Transporter User Guide*, B035-2445.

where:

Option	Specifies
<i>jobName</i>	A unique name that identifies the load job. Recommendation: Teradata strongly recommends that you specify names for your jobs when multiple load jobs are running simultaneously.

## tdload and the Teradata PT Protocols

The **tdload** command has the intelligence to determine the most appropriate load operator (for example, Load, Update, Stream or Inserter) or export operator (for example, Export) to be used for the Easy Loader job based on the status of the target table. For example,

- If the target table is empty and has no secondary indexes, **tdload** knows to use the Load operator to load data.
- If the target table is not empty, **tdload** may use the Update operator to load data.
- If a value is specified for either the --SourceTable option or --SelectStmt option, **tdload** knows to use the Export operator to extract data from database tables.

### Note:

Any attributes not supported as part of the allowed set of **tdload** command line options must be specified in the job variables file. Because **tdload** internally calls **tbuild**, these attributes get passed as part of the job variables file to the **tbuild** command.

If values are specified for both the --SourceTable option and the --SelectStmt option, **tdload** ignores the value of --SourceTable option. A warning message is returned to the console and the job continues with the schema inferred from the SELECT statement.

## Defining a Flat File Source Table

To specify a flat file source table for **tdload**, specify either the -f option or the --SourceFileName option.

The following combination of options are mutually exclusive:

- -f and --SourceTable
- -f and --SelectStmt

An error is returned if either one of these combinations is specified on the **tdload** command line or in the job variables file.

## Defining a Source Table

To specify a source table for **tdload**, you must specify either the --SourceTable option or the --SelectStmt option and the following job options on the command line or in the job variables file:

- --SourceTpId
- --SourceUserName
- --SourceUserPassword

You are not required to specify any of the following job options (either on the command line or in the job variables file) when defining a data source:

- --SourceAccountId
- --SourceWorkingDatabase
- --SourceMaxSessions
- --SourceMinSessions
- --SourceInstances

**Note:**

If both the SourceTable option and the --SelectStmt option are defined, the --SourceTable option is ignored and a warning message is returned to the console.

## Inferring the Data Schema

- When moving data from a flat file to a database table, the data schema is inferred from the target table.
- When moving data between database tables, the data schema is inferred from either:
  - The Source table specified in the --SourceTable option, or
  - The SQL SELECT statement defined in the --SelectStmt option.

An error is returned if no target table is specified.

If the source data does not match the inferred schema, the Easy Loader job terminates with an error.

## tdload and Checkpoint Files

Teradata PT automatically deletes the checkpoint files of successful jobs, but retains the checkpoint files of unsuccessful jobs. If a previously run job is unsuccessful, you can either let the checkpoint files be reused by that failed job or delete them so they are not picked up by any other job executed under the same login name.

To avoid problems, you must either carefully maintain the checkpoint files of unsuccessful jobs, or, preferably, always specify a job name using the *jobname* option.

## tdlog

### Purpose

The tdlog command is used to extract the logs produced by a running job.

## Syntax

The Teradata PT command, `tdlog`, has the following syntax:

`tdlog — jobId` ➔

where:

Syntax Element	Description
<code>jobId</code>	Unique job identifier. This value is provided in the console output generated by the <code>tdload</code> command. The <code>tdlog</code> command <code>tdlog jobId</code> is equivalent to <code>tlogview -j jobId-f "*" -g</code> . Either command generates the contents of all private logs. Also, the purpose of the command is to obtain the contents of the log of running and terminated jobs.

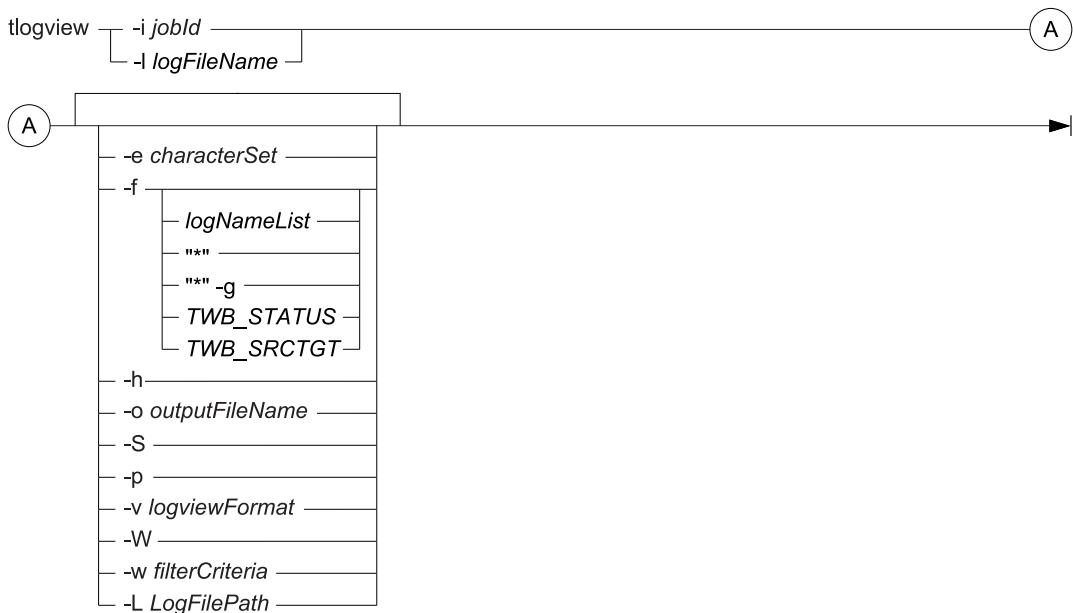
## tlogview

### Purpose

The `tlogview` command displays the contents of the log files produced from running a Teradata PT job. All logs contain a banner that displays the version number of the active database for diagnostic purposes.

## Syntax

The order of the options is not important.



where:

Syntax Element	Description
<code>-e characterSet</code>	Option to specify that the tlogview output, including tlogview error messages, will be displayed in UTF-16. The only valid value for <code>characterSet</code> is UTF16 (no hyphen). If the <code>-e characterSet</code> is not specified, <b>tlogview</b> defaults to displaying in UTF-8.
<code>-f **</code>	Option that requests the output of the public log and all private logs.
<code>-f ** -g</code>	Option that sorts the output of each private log separately, ordered by logname. The <code>-g</code> attribute must be used with <code>**</code> , as shown in the following example:  <code>tlogview -j JobID -f ** -g</code>  This command gets all the public and private logs, grouped by log name.  <b>Note:</b> If the TraceLevel attribute is operational, the sort is done by operator instance.
<code>-f TWB_STATUS</code>	Option that identifies metadata, query job statistics, and status. If this option is not specified, no metadata, statistics, or status are identified.
<code>-f TWB_SRCTGT</code>	Option that identifies metadata, the query source, and target information. If this option is not specified, no metadata, source, or target are identified.
<code>-h</code>	Option that requests the job start time. If this option is specified, tlogview prints the job start time as the first line on the output.
<code>-j jobId</code>	Option that identifies the log output from a running or terminated Teradata PT job.

Syntax Element	Description
	<p>Use this option when viewing the log identified by <i>jobId</i>. The job can be running or terminated.</p> <ul style="list-style-type: none"> <li>When the -W option is not used with this option for a running job, <code>tlogview</code> displays the log information produced to that point and terminates.</li> <li>When the -W option is used with this option, <code>tlogview</code> displays the log information produced to that point, then it waits for more log information to be produced by the job, terminating when the target job terminates.</li> </ul> <p>If the job is already terminated, <code>tlogview</code> displays the log information and terminates. In general, always use the -j option unless you move or rename the log file. In this case, use the -l option and specify the new location.</p> <p>The -j and -l options are mutually exclusive.</p>
<code>-l logFileName</code>	<p>Option that identifies a physical log file.</p> <p>Use this option to view the log identified by a physical file name from a Teradata PT job. Log files are usually named with extensions of .out.</p> <p>The -j and -l options are mutually exclusive.</p>
<code>-L logFilePath</code>	<p>Option that helps specify the path to the physical log file.</p> <p>If -L is not specified, <code>tlogview</code> will look for the physical log file in the TPT logs directory when used with -j jobId. If -L is not specified, <code>tlogview</code> will look in the current directory when used with -l logFileName.</p> <p>If -L is specified, logFilePath will override the default logs location when used with -j jobId or -l logFileName.</p>
<code>-f logNameList</code>	<p>Option that identifies a “from list,” that describes what log reports to view.</p> <p>The default, if nothing is specified, is to view only public log reports. If both private and public log reports are requested, the “from list” must include the term “public” in the list. Listed items must be separated by commas, as shown in the following example:</p> <pre>tlogview -j jobID -f PUBLIC,privateLog1,privateLog2</pre> <p>This command gets all the public logs and private logs 1 and 2.</p>
<code>-o outputFileName</code>	<p>Option that designates a specific output file.</p> <p>If this option is specified, <code>tlogview</code> writes the output to the specified file. If this option is not specified, output is written to standard output.</p>
<code>-p</code>	Option that lists the names of the private log reports contained in a target job log.
<code>-v logviewFormat</code>	<p>Option that specifies the log view format.</p> <p>Use the following values to specify the output format of log messages:</p> <ul style="list-style-type: none"> <li>%A for Task ID</li> <li>%D for Date Time</li> <li>%E for Destination</li> <li>%G for Message Catalog</li> <li>%L for Delta Time</li> <li>%M for Message Text</li> <li>%N for Node Name</li> </ul>

Syntax Element	Description
	<ul style="list-style-type: none"> <li>• %P for Process ID</li> <li>• %T for Task Name</li> <li>• %U for Message Number</li> </ul> <p>For example, the following command displays log messages with time stamps, task names, and message texts separated by “-” characters:</p> <pre>tlogview -j &lt;jobID&gt; -v "%D-%T-%M"</pre>
-W	Option used in conjunction with the <i>-j jobID</i> option to tell <i>tlogview</i> to continue displaying additional log information until the target job terminates.
-w <i>filterCriteria</i>	<p>Option that specifies the filter criteria.</p> <p>The filter criteria allows <i>tlogview</i> to do filtering on the log messages. Only those log messages satisfying the filter criteria are output from <i>tlogview</i>.</p> <p>Without the <i>-w</i> option, <i>tlogview</i> selects all messages in the public and private logs.</p> <p>For example, the following command displays log messages where Task Name equals “SELECT_2[0001]”.</p> <pre>tlogview -j &lt;jobID&gt; -w ‘TASKNAME=”SELECT_2[0001]”’</pre>

## twbcmd

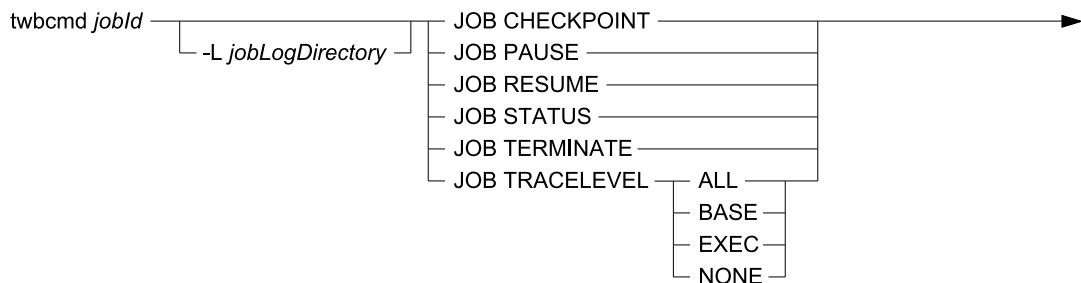
### Purpose

There are two types of commands available through **twbcmd**, the External Command Interface.

- **Job-level commands:** these can be used to:
  - Create checkpoints
  - Pause and then resume a job
  - Retrieve job status
  - Terminate a job
  - Enable job tracing at any time during job execution
- **Operator-level commands:** these can be used to:
  - Set the statement rate for the Stream operator within the job
  - Set the periodicity for the Stream operator within a job

### Syntax for Job-level Commands

The following syntax is required for job-level commands using **twbcmd**.

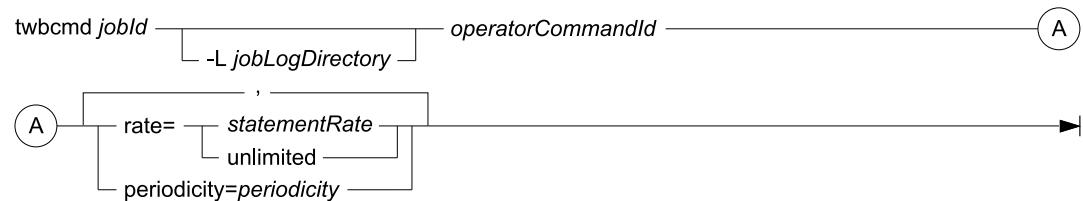


where:

Syntax Element	Description
<i>jobId</i>	<i>jobId</i> is the <i>jobName</i> followed by a “-”, then followed by the Teradata PT-generated sequence number. <i>jobName</i> specifies the name of the target job.
- L <i>jobLogDirectory</i>	Option that designates the location of the Teradata PT files created during job execution. This option is only applicable when - L <i>jobLogDirectory</i> is specified in the <b>tbuild</b> or <b>tdload</b> command. <i>jobLogDirectory</i> is the full path name of the directory in which the Teradata PT files are stored. This option is not supported on z/OS.
JOB CHECKPOINT	Takes an immediate checkpoint, then continues the job.
JOB PAUSE	Takes an immediate checkpoint, then suspends processing.
JOB RESUME	Resumes a paused job.
JOB STATUS	Writes a status record for each active operator instance to the TWB_STATUS log, and displays row processing statistics while continuing the job.
JOB TERMINATE	Takes an immediate checkpoint, then terminates the job. The job retains the checkpoint files, and is therefore restartable.
JOB TRACELEVEL	Enables job tracing at any time during job execution. One of the following arguments is required when enabling job tracing: <ul style="list-style-type: none"><li>• ALL</li><li>• BASE</li><li>• EXEC</li><li>• NONE</li></ul>
ALL	Enables both BASE and EXEC job tracing during job execution.
BASE	Enables job tracing of all job processes during job execution.
EXEC	Enables execution tracing of all operator processes during job execution.
NONE	Disables job tracing during job execution.

## Syntax for Operator-Level Commands

The following syntax is required for operator-level commands using twbcmd.



where:

Syntax Element	Description
<i>jobId</i>	<i>jobId</i> is the <i>jobName</i> followed by a “-”, then followed by the Teradata PT-generated sequence number. <i>jobName</i> specifies the name of the job targeted by the command.
<i>-L jobLogDirectory</i>	Option that designates the location of the Teradata PT files created during job execution. This option is only applicable when <i>-L jobLogDirectory</i> is specified in the tbuild or tdload command. <i>jobLogDirectory</i> is the full path name of the directory in which the Teradata PT files are stored. This option is not supported on z/OS.
<i>operatorCommandId</i>	Specifies the identity of a Stream operator copy and allows changes to the Rate value for that copy only, while the job is running. When a job step contains multiple Stream operator copies with differing rate values, Teradata PT will automatically use the lowest rate value for all instances. Use the operatorCommandID to identify an operator copy and change the rate value.

Syntax Element	Description
	<p><b>Note:</b></p> <p>To change the Rate value of a Stream operator copy, you must declare the operatorCommandID attribute in the Stream operator DEFINE OPERATOR statement.</p> <p>Use one of the following two methods to change the Rate:</p> <ul style="list-style-type: none"> <li>Assign a value such as 'step#1' to the OperatorCommandID attribute in the Stream operator copy that requires the Rate change (in the APPLY statement). Then use the following command:</li> </ul> <pre>twbcmd &lt;jobid&gt; step#1 RATE=2000</pre> <ul style="list-style-type: none"> <li>If a value was NOT assigned for the OperatorCommandID attribute in the Stream operator copy that requires the Rate change, the system will generate a default ID composed of the operatorName and a process ID number. Process ID numbers for operators appear on the command screen when the job step that contains the operator begins to execute. An example of a system-generated ID would be similar to the following: <i>my_operator_name3456</i></li> </ul> <p>Use the following command to identify an operator by default ID and change the rate value:</p> <pre>twbcmd &lt;jobid&gt; my_operator_name3456 RATE=2000</pre>
rate= <i>statementRate</i>	<p>Option that specifies the maximum number of DML statements per minute the Stream operator can submit to the database.</p> <p>Use the <b>twbcmd</b> Rate option to slow down a Teradata PT job for other higher priority jobs and to speed it up again after the priority job has completed.</p> <p>The statement rate can also be specified in a DEFINE OPERATOR statement, by using the Stream operator Rate attribute. When both values are present, the <i>twbcmd</i> rate value will supersede the Stream operator attribute Rate value.</p> <p><b>Note:</b></p> <p>When a job step contains multiple occurrences of the Stream operator with differing Rate values, Teradata PT will automatically use the lowest rate value for all instances.</p> <p>The specified Rate value must be either:</p> <ul style="list-style-type: none"> <li>a whole number greater than zero or,</li> <li>unlimited</li> </ul> <p><b>Note:</b></p> <p>The default statement rate, if not set using either the Stream operator Rate attribute or by <b>twbcmd</b>, is unlimited. Specifying 'unlimited' for the <b>twbcmd</b> Rate value means you are changing the value back to the default after having set the value in the Stream operator.</p> <p>When the <b>twbcmd</b> Rate option is used, the Stream operator changes the statement rate to the new value and displays a message showing the new value. If the specified rate is greater than the packing factor, the Stream operator will send the number of rows equal to the packing factor.</p>

Syntax Element	Description
periodicity= <i>periodicity</i>	<p>Option that specifies that the DML statements sent by the Stream operator to the database be as evenly distributed as possible over each one minute interval. <i>periodicity</i> sets the number of sub-intervals per minute.</p> <p>For instance, if the rate is 1600 and the periodicity is 10, then the maximum number of statements submitted is 160 (1600/10) every 6 (60/10) seconds. Valid values are between 1 and 600.</p> <p>The default value is 4, that is, four 15 second intervals per minute.</p> <p>If the statement rate is unlimited, then <i>periodicity</i> will be ignored.</p> <p><b>Note:</b></p> <p>The periodicity can also be specified in a DEFINE OPERATOR statement, by using the Stream operator Periodicity attribute. When both values are present, <i>periodicity</i> will supersede the Stream operator Periodicity attribute value.</p>

## Usage Notes

Use the following **twbcmd** syntax examples for commands on all platforms except z/OS.

- For job-level commands:

```
twbcmd <jobid> <command>
```

where *jobId* is the identification of the target job, and *command* is the name of the external command that you want to execute.

- For operator-level commands:

```
twbcmd <jobid> <operator command Id> <attribute name>=<value>
```

or, where more than one attribute is specified,

```
twbcmd <jobid> <operator command Id> <attribute name>=<value>,
<attribute name>=<value>
```

where *jobId* is the name of the target job, *operatorCommandId* is the identifier of a Stream operator within that job, *attributeName* is the attribute you want to employ, and *value* is the specified rate or periodicity.

On z/OS, use MODIFY syntax as follows.

- For job-level commands:

```
F <jobid>,APPL=<command>
```

Quotation marks are not necessary around the command name. If quotes are present, they are sent as part of the command.

- For operator-level commands:

F <jobid>,APPL=<operator name> <attribute name>=<value>

or, where more than one attribute is specified,

F <jobid>,APPL=<operator name> <attribute name>=<value>, <attribute>=<value>

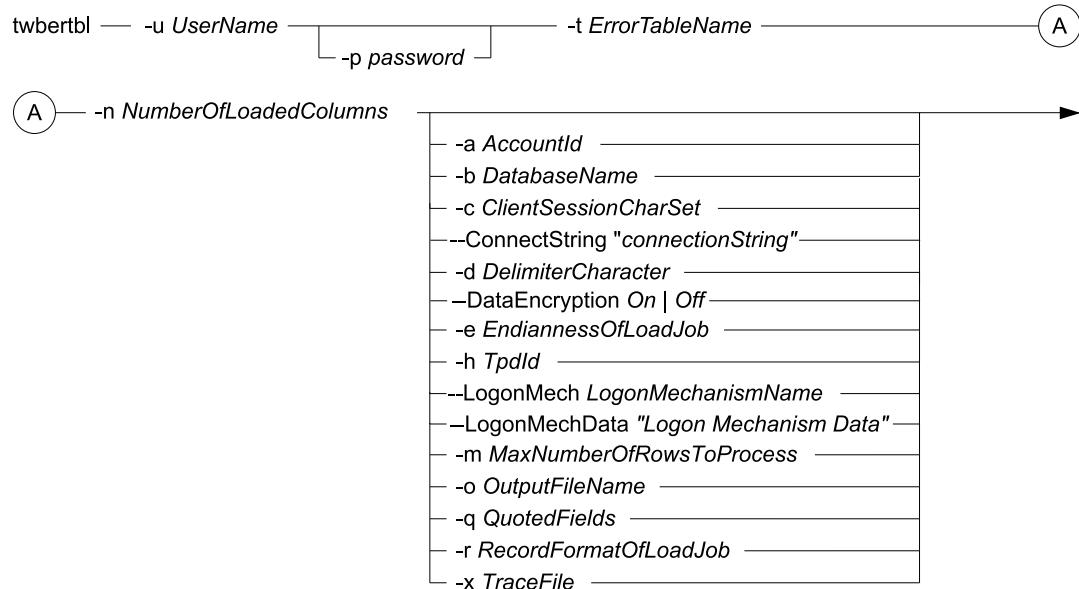
twbertbl

## Purpose

The Teradata PT Error Table Extractor command, `twbertbl`, extracts the error information from the acquisition phase error table or the TPT Stream Operator error table.

The Teradata PT Error Table Extractor can be used to extract the error information to a data file. The user can correct the data in the data file and reload the data.

## Syntax



where:

**Note:**

All values for the options are case-insensitive unless otherwise specified.

Syntax Element	Description
<code>-u <i>UserName</i></code>	User name of the Teradata logon account.
<code>-p <i>Password</i></code>	Password of the Teradata logon account. If omitted, the twbertbl tool will prompt you for the password on non-z/OS platforms. If omitted, the twbertbl tool will terminate with an error on z/OS platforms. The password is case sensitive
<code>-t <i>ErrorTableName</i></code>	Name of the error table. The error table name can be fully qualified.
<code>-n <i>NumberOfLoadedColumns</i></code>	Number of columns sent to the database in the original load job. The -n option is required for the DELIMITED and TEXT record formats. The -n option is not required for the BINARY, FORMATTED, and UNFORMATTED record formats.
<code>-h <i>TdpId</i></code>	TdpId of the database.
<code>-a <i>AccountId</i></code>	Account id of the logon account.
<code>-b <i>DatabaseName</i></code>	Database name for the error table. If the database name is specified, the twbertbl tool prepends the database name to the error table name.
<code>-c <i>ClientSessionCharSet</i></code>	Client session character set. The default is 'ASCII' on non-z/OS platforms. The default is 'EBCDIC' on z/OS platforms.
<code>-o <i>OutputFileName</i></code>	Name of the output file that stores the error table information. The default is < <i>ErrorTableName</i> >.txt. The output file name is case-sensitive.
<code>-m <i>MaxNumberOfRowsToProcess</i></code>	Maximum number of rows to write to the output file. The default is unlimited.
<code>-d <i>DelimiterCharacter</i></code>	Single-byte or multi-byte characters delimiter used to separate fields in the output file. The default is the pipe character ' '. The delimiter characters are case-sensitive. The -d option is applicable when the record format is DELIMITED. The -d option is not applicable when the record format is BINARY, FORMATTED, TEXT, or UNFORMATTED.
<code>-q <i>QuotedFields</i></code>	Option to quote the fields in the output file.

Syntax Element	Description
	<p>Value can be 'Y' or 'N'. The default is 'N'. If the value is 'Y', all fields will be quoted. The -q option is applicable when the record format is DELIMITED. The -q option is not applicable when the record format is BINARY, FORMATTED, TEXT, or UNFORMATTED.</p>
<code>-e EndiannessOfLoadJob</code>	<p>Endianness of the error data in the error table. Value can be 'BE' or 'LE'. 'BE' means big-endian. 'LE' means little-endian. If omitted, the twbertbl tool will attempt to determine the endianness of the error data in the error table.</p>
<code>-r RecordFormat</code>	<p>Record format of the error data in the error table. Value can be 'DELIMITED', 'BINARY', 'FORMATTED', 'TEXT', or 'UNFORMATTED'. The default is 'DELIMITED'.</p>
<code>-x TraceFile</code>	<p>Name of the trace file for the twbertbl tool. By default, trace is off.</p>
<code>--LogonMech LogonMechanismName</code>	<p>Specifies which logon mechanism to use. The job terminates if the value exceeds 8 bytes. For information on specification requirements for LogonMech, see "Logon Security" in <i>Teradata® Parallel Transporter User Guide</i>, B035-2445.</p>
<code>--LogonMechData "Logon Mechanism Data"</code>	<p>Specifies additional logon data for external authentication. The logon mechanism data is case-insensitive. Specification of this option is required for some external authentication methods. For information on specification requirements for LogonMechData, see "Logon Security" in <i>Teradata® Parallel Transporter User Guide</i>, B035-2445.</p>
<code>--ConnectionString "connectionString"</code>	<p>Specifies the connection string. The connection string will be passed to CLI. CLI will validate the connection string. For information on connection string, see <i>Teradata® Call-Level Interface Version 2 Reference for Workstation-Attached Systems</i>, B035-2418.</p> <p><b>Note:</b> The TPT Connection String feature is available on all platforms, except for the TDP variant of TPT on z/OS.</p>
<code>--DataEncryption On   Off</code>	<p>Enables full encryption of SQL requests, responses and data. Valid values are:</p> <ul style="list-style-type: none"> <li>• On = All SQL requests, responses, and data are encrypted.</li> <li>• Off = No encryption occurs (default).</li> </ul>

Syntax Element	Description
	<p><b>Note:</b> This option does not include logon encryption.</p>

## Usage Notes

If the -q option is set to 'Y', all fields will be quoted. Here is an example of the output file:

```
"EMP_NUM" | "2683" | "100" | ""128""
```

When the record format is BINARY, FORMATTED, TEXT, or UNFORMATTED, only the error data parcel or error host data will be extracted. The error code, error field, import sequence number, DML sequence number, statement sequence number, apply sequence number, and row sequence number will not be extracted.

## Examples – twbertbl

### Example 1

This example extracts the error information from the Load operator's acquisition phase error table. This example uses the sample schema for the Load operator job:

```
EMP_ID    VARCHAR(20),
EMP_NUM   VARCHAR(10)
```

This example uses this sample content for the Load operator's acquisition phase error table:

ErrorCode	ErrorFieldName	DataParcel
-----	-----	-----
2683	EMP_NUM	0003003130300300313238

- Use the following twbertbl syntax example for commands on all platforms except z/OS:

```
twbertbl -h <TdpId> -u <UserName> -t <ErrorTableName>
         -n <NumberOfLoadedColumns> -o <OutputFileName>
```

- On z/OS, use the following syntax example inside a JCL:

```
//TWBERTBL EXEC PGM=TWBERTBL,
// PARAM=' -h TDP0 -u MyUser -p MyPass -n 2 -t LoadOpErrTbl1
//                   -o DD:DATA'
```

- After the extraction, the contents of the output file name will look like this, separated by a delimiter character:

```
EMP_NUM|2683|100|128
```

The first field in the output file name is the field name that caused each error.

The second field is the error code for each error.

The rest of the fields are the column values of the data records for each error.

### Example 2

This example extracts the error information from the Update operator's acquisition phase error table.

This example uses this sample schema for the Update operator job:

```
EMP_ID    VARCHAR(20),
EMP_NUM   VARCHAR(10)
```

This example uses this sample content for the Update operator's acquisition phase error table:

```
ImportSeq DMLSeq SMTSeq ApplySeq SourceSeq
-----
1       1       1       1       1

ErrorCode ErrorField HostData
-----
2683     EMP_NUM    0003003130300300313238
```

- Use the following twbertbl syntax example for commands on all platforms except z/OS:

```
twbertbl -h <TdpId> -u <UserName> -t <ErrorTableName>
         -n <NumberOfLoadedColumns> -o <OutputFileName>
```

- On z/OS, use the following syntax example inside a JCL:

```
//TWBERTBL EXEC PGM=TWBERTBL,
// PARAM=' -h TDP0 -u MyUser -p MyPass -n 2 -t UpdateOpErrTbl1
//           -o DD:DATA'
```

- After the extraction, the contents of the output file name will look like this, separated by a delimiter character:

```
1|1|1|1|1|EMP_NUM|2683|100|128
```

The first field in the output file name is the import sequence number that caused each error.

The second field is the DML sequence number that caused each error.

The third field is the statement sequence number that caused each error.

The fourth field is the apply sequence number that caused each error.

The fifth field is the row sequence number that caused each error.

The sixth field is the error code for each error.

The seventh field is the field name that caused each error.

The rest of the fields are the column values of the data records for each error.

### Example 3

This example extracts the error information from the Load operator's acquisition phase error table using Multi-byte character delimiter.

This example uses this sample schema for the Load operator job:

```
EMP_ID    VARCHAR(20),
EMP_NUM   VARCHAR(10)
```

This example uses this sample content for the Load operator's acquisition phase error table:

ErrorCode	ErrorFieldName	DataParcel
2683	EMP_NUM	0003003130300300313238

- Use the following twbertbl syntax example for commands on all platforms except z/OS:

```
twbertbl -h <TdpId> -u <UserName> -t <ErrorTableName>
         -n <NumberOfLoadedColumns> -o <OutputFileName> -d <Multi-byte
         characters delimiter>
```

- On z/OS, use the following syntax example inside a JCL:

```
//TWBERTBL EXEC PGM=TWBERTBL,
// PARAM='-h TDP0 -u MyUser -p MyPass -n 2 -t Update0pErrTbl1
// -o DD:DATA -d aaa'
```

- After the extraction, the contents of the output file name will look like this, separated by multi-byte characters delimiter 'aaa':

```
EMP_NUMaaa2683aaa100aaa128
```

The first field in the output file name is the field name that caused each error.

The second field is the error code for each error.

The rest of the fields are the column values of the data records for each error.

### Example 4

This example extracts the error information from the Stream operator's error table.

This example uses this sample schema for the Stream operator job:

```
EMP_ID VARCHAR(20),
EMP_NUM VARCHAR(10)
```

This example uses this sample content for the Stream operator's error table:

LOADSTARTTIME	DMLSEQ	SMTSEQ	SOURCESEQ	DATASEQ	ERRORCODE
2018-01-24 17:35:26.000000	1	1	1	1	2617
<b>ERRORMSG</b>					
Overflow occurred computing an expression involving STREAM_TARGET_EMP_TABLE.COL2					
<b>ERRORFIELD HOSTDATA</b>					
0	000B00202020202020202020310A00313238202020202020				
<b>ROWINSERTTIME</b>					
2018-01-24 17:35:29.360000					

- Use the following twbertbl syntax example for commands on all platforms except z/OS:

```
twbertbl -h <TdpId> -u <UserName> -t <ErrorTableName>
         -n <NumberOfLoadedColumns> -o <OutputFileName>
```

- On z/OS, use the following syntax example inside a JCL:

```
//TWBERTBL EXEC PGM=TWBERTBL,
// PARAM=' -h TDP0 -u MyUser -p MyPass -n 2 -t StreamOpErrTbl
//           -o DD:DATA'
```

- After the extraction, the contents of the output file name will look like this, separated by a delimiter character:

```
2018-01-24 17:35:26.000000|2018-01-24 17:35:29.360000|1|1|1|1|0|2617|
Overflow occurred computing an expression involving
STREAM_TARGET_EMP_TABLE.COL2| 1|128
```

The first field in the output file name is the Queue Insertion TimeStamp (QITS). The value indicates when the job started. On restart, it indicates when the job restarted.

The second field indicates when the row was inserted into the Stream operator error table.

The third field is sequence number assigned to the DML group in which the error occurred.

The fourth field is the sequence number of the DML statement within the DML group (as indicated by the previous column DMLSeq) being executed when the error occurred.

The fifth field is the sequence number assigned to the row from the input source (the DataSeq number) in which the error occurred.

The sixth field is the sequence number assigned to the input source in which the error occurred.

The seventh field is the field that caused the error. This field is always 0, because, currently, the database does not have a way to tell the TPT Stream operator which field in the data error record caused the error. However, the error message can indicate the column name that caused the data error.

The eighth field is the error code.

The ninth field is the error message.

The rest of the fields are the column values of the data records for each error.

## Restrictions and Limitations

- The Teradata PT Error Table Extractor supports VARBYTE or BLOB for the DATAPARCEL column in the FastLoad acquisition phase error table.  
The Teradata PT Error Table Extractor supports VARBYTE or BLOB for the HOSTDATA column in the MultiLoad acquisition phase error table.  
The Teradata PT Error Table Extractor supports VARBYTE or BLOB for the HOSTDATA column in the Stream operator error table.  
The DATAPARCEL or HOSTDATA column is VARBYTE when the TPT Load or Update operator job is running against a database that does not have the Vantage 1MB Permanent Row feature enabled.  
The HOSTDATA column is VARBYTE when the schema for the TPT Stream operator job is less than 64K.  
The DATAPARCEL or HOSTDATA column is BLOB when the TPT Load or Update operator job is running against a database that has the Vantage 1MB Permanent Row feature enabled.  
The HOSTDATA column is BLOB when the schema for the TPT Stream operator job is greater than 64K.  
If the DATAPARCEL or HOSTDATA column is not VARBYTE or BLOB, the Teradata PT Error Table Extractor will terminate the job with an error message.
- The Teradata PT Error Table Extractor supports the acquisition phase error table for only these Teradata PT operators:
  - Load operator:
  - Update operator:
  - Stream operator

- The schema for the Teradata PT Load, Stream and Update operator jobs must have all VARCHAR columns when the record format is DELIMITED. For other record formats, the Load, Stream and Update jobs can have different column data types.

- The Teradata PT Error Table Extractor does not support the extraction of the application phase error tables.

- On z/OS, the Teradata PT Error Table Extractor does not support the extraction of the acquisition phase error tables which were created on non-z/OS platforms, because the character encoding on z/OS is different from non-z/OS platforms.

On z/OS, the Teradata PT Error Table Extractor can only support the extraction of the acquisition phase error tables that were created from jobs run from the z/OS platforms.

- On non-z/OS platforms, the Teradata PT Error Table Extractor does not support the extraction of the acquisition phase error tables which were created on z/OS, because the character encoding on non-z/OS is different from z/OS platforms.

On non-z/OS platforms, the Teradata PT Error Table Extractor can only support the extraction of the acquisition phase error tables that were created from jobs run from the non-z/OS platform.

- On z/OS, the Teradata PT Error Table Extractor does not prompt the user for the password if the -u Password option is not specified. You must specify the -u Password option on z/OS.
- The Teradata PT Error Table Extractor is not certified to extract the error information from the acquisition phase error table for these Teradata utilities.

– FastLoad

– MultiLoad

– TPump

However, the Teradata PT Error Table Extractor can extract the error information from the acquisition phase error table for the Teradata FastLoad and MultiLoad utilities if the data for the FastLoad or MultiLoad job was sent in the indicator mode. If the data was sent in the non-indicator mode, then the Teradata PT Error Table Extractor cannot extract the error information.

The Teradata PT Error Table Extractor cannot extract the error information from a TPump error table, because the Teradata PT Error Table Extractor does not know the layout of the TPump error table.

## **twbkill**

### **Purpose**

The **twbkill** command terminates all tasks running as part of an application.

When you start Teradata PT using the **tbuild** command, Teradata PT automatically starts tasks that run in the background. Using the **twbkill** command with the name of the job terminates all tasks running as part of the application.

**Note:**

On UNIX systems, the **twbkill** user must either be superuser or have the same user ID as the target Teradata PT job. This command is not available on z/OS systems.

**Syntax**

```
twbkill — jobId —►
          | -L jobLogDirectory —►
```

where:

Syntax Element	Description
<i>jobId</i>	Job ID. To run multiple jobs concurrently, use a unique job name to qualify each job, thus avoiding the generation of duplicate names for checkpoint files. For more information, see “Setting Checkpoint Options” and “Restarting a Job From the Last Checkpoint Taken,” in the <i>Teradata Parallel Transporter User Guide</i> (B035-2445).
- L <i>jobLogDirectory</i>	Option that designates the location of the Teradata PT files created during job execution. This option is only applicable when - L <i>jobLogDirectory</i> is specified in the <b>tbuild</b> or <b>tdload</b> command. <i>jobLogDirectory</i> is the full path name of the directory in which the Teradata PT files are stored. This option is not supported on z/OS.

**Example – twbkill**

Use the following twbkill command:

```
twbkill wilson-235
```

To terminate all tasks in the designated Teradata PT job. An error message results if the termination is not successful. This command creates the following output:

```
# twbkill
Using job directory /home/wilson/jobs
wilson-235 killed
```

## twbrmcp

### Purpose

The **twbrmcp** command removes all checkpoint files associated with *userId* or *jobId*.

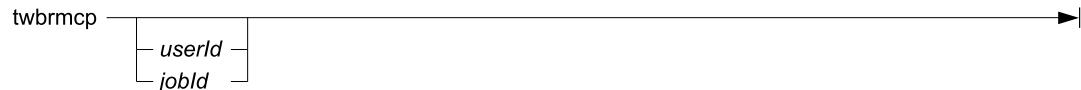
For information on checkpointing, see the *Teradata Parallel Transporter User Guide* (B035-2445).

In the event of a job failure, the checkpoint files for the job are retained in the checkpoint directory to support job restart from the last checkpoint taken. If the job needs to restart from the beginning, then the checkpoint file(s) for that job must be removed.

The **twbrmcp** command specifies which checkpoint files to remove by requiring the use of either the name of the job associated with the checkpoint files or the *userId* under which the job was executed.

### Syntax

Specify one of the two available options to select the files to be removed.



where:

Syntax Element	Description
<i>userId</i>	Option to remove all checkpoint files associated with the logon <i>userId</i> .
<i>jobId</i>	Option to remove all checkpoint files associated with the jobname specified in the <b>tbuild</b> command for a job.

### Example – twbrmcp

The following examples use the **twbrmcp** command.

- The following **twbrmcp** command removes checkpoint files associated with a job and returns output verifying the removal:

```

$ twbrmcp MyJob
Checkpoint file /opt/tbuild/client/<version>/tbuild/checkpoint/
MyJobLVCP has been removed
Checkpoint file /opt/tbuild/client/<version>/tbuild/checkpoint/
MyJobCPD1 has been removed

```

```
Checkpoint file /opt/tbuild/client/<version>/tbuild/checkpoint/
MyJobCPD1 has been removed
```

- The following **twbrmcp** command removes checkpoint files associated with a user and returns output verifying the removal:

```
$ twbrmcp ay160001
Checkpoint file /opt/tbuild/client/<version>/tbuild/checkpoint/
ay160001LVCP has been removed
Checkpoint file /opt/tbuild/client/<version>/tbuild/checkpoint/
ay160001CPD1 has been removed
Checkpoint file /opt/tbuild/client/<version>/tbuild/checkpoint/
ay160001CPD1 has been removed
```

- The following **twbrmcp** command, used without an attribute, returns information on what parameters are required to designate a checkpoint file for removal.

```
cs4400s3:/ > twbrmcp
Usage: twbrmcp LogonID
       twbrmcp JobName
Parameter description:
LogonID   The user logon ID
JobName   The job name specified for the 'tbuild' command
```

#### Note:

The **twbrmcp** command is not available on z/OS.

## **twbstat**

### Purpose

The **twbstat** command displays the names of currently active Teradata PT jobs.

This command is not available on z/OS systems.

### Syntax

```
twbstat [ -L jobLogDirectory ] ➤
```

where:

Syntax Element	Description
- L <i>jobLogDirectory</i>	<p>Option that designates the location of the Teradata PT files created during job execution.</p> <p>This option is only applicable when - L <i>jobLogDirectory</i> is specified in the <b>tbuild</b> or <b>tdload</b> command. <i>jobLogDirectory</i> is the full path name of the directory in which the Teradata PT files are stored.</p> <p>This option is not supported on z/OS.</p>

## Example – twbstat

The following example uses the twbstat command. It returns a list of the Teradata PT jobs currently running on the system.

The following twbstat command:

```
#twbstat
```

Creates the following output:

```
Using job directory/home/c1151001/jobs
Jobs running: 3
c1151001-112
lol142000-133
dcc13370-147
```

# Object Definitions and the APPLY Statement

## Overview

The following topics describe Teradata PT:

- **Object definition statements:** These make up the declarative section of a Teradata PT job script and are used to perform the job tasks.
- **Executable statement:** The APPLY statement makes up the executable section of a Teradata PT job and is used to execute a Teradata PT job.

Topics include:

- [Object Definition Statements](#)
- [Syntax for Attribute Declarations](#)
- Descriptions of the following object definition statements:
  - [DEFINE JOB](#)
  - [DEFINE SCHEMA](#)
  - [DEFINE OPERATOR](#)
- [APPLY](#) statement

For information about Teradata PT reserved keywords, see [Restricted Words](#).

## Object Definition Statements

Object definition statements define all the Teradata PT objects that are used to carry out the objectives of the job in a script.

Teradata PT objects must be defined before they are used by the APPLY statement to execute a Teradata PT job.

The following table provides a brief description of the function of each object definition statement.

To access the detailed information on each statement, click the links.

**Teradata PT Object Definitions Statements**

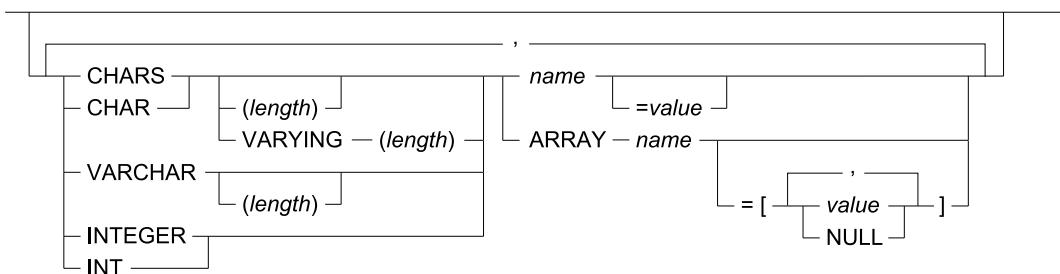
Statement	Function
<a href="#">DEFINE JOB</a>	Required. Packages all define and apply statements. Names and defines the job. Teradata PT objects are first defined, then referenced within the job definition.
<a href="#">DEFINE OPERATOR</a>	Required. Defines the properties of a specific Teradata PT operator.

Statement	Function
<a href="#">DEFINE SCHEMA</a>	<p>Required.</p> <p>Defines the structure, or composition of an abstract data object in terms of columns of specific data types.</p> <p>Schemas can be used to describe the structure of actual data objects such as files or tables. A given schema definition can be used to describe multiple data objects.</p>

## Syntax for Attribute Declarations

The following figure shows the syntax for operator attribute declarations.

For operator-specific syntax, see the topics on that operator in this document.



where:

Syntax Element	Description
ARRAY	Keyword specifying a multivalued, or array, attribute. The ARRAY keyword is optional when character strings are assigned to the array attribute of the operator to be defined or in the APPLY statement.
CHARS CHAR	Keyword specifying either a fixed- or variable-length character data type.
INTEGER INT	Keyword specifying a four-byte integer numeric data type.
<i>length</i>	Length specification for non-numeric data types. The length specification is: <ul style="list-style-type: none"> <li>Required for column definition of variable-length data types.</li> <li>Optional for attribute declaration of variable-length data types.</li> </ul>
NULL	Keyword in the initial values list for an array attribute indicating, that no initial value is being specified for that array element.
<i>value</i>	Optional value in an attribute declaration list. The value can be a: <ul style="list-style-type: none"> <li>Character string</li> <li>Integer</li> <li>Job variable reference, for example @job_attr_name.</li> </ul>

Syntax Element	Description
VARCHAR VARYING	Keyword specifying a variable-length character string data type.

## DEFINE JOB

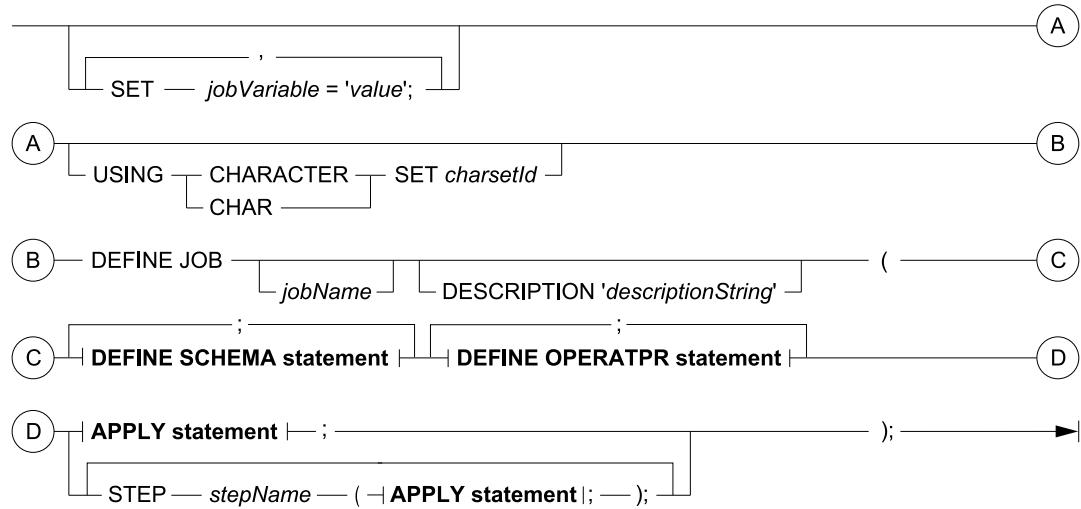
### Purpose

The DEFINE JOB statement defines a Teradata PT job.

All Teradata PT objects must first be defined before they can be referenced in a job definition.

### Syntax

The DEFINE JOB statement syntax order is important.



where:

Syntax Element	Description
DEFINE JOB	Required keyword phrase specifying the beginning of the job definition.
DESCRIPTION ' <i>descriptionString</i> '	Optional keyword phrase providing a descriptive comment about the defined job object.
<i>jobName</i>	Required internal Teradata PT metadata name of the defined job object. The name does not have to be the same as the file name of the script.
SECONDS SEC	Optional keyword indicating that the preceding value is expressed in seconds.

Syntax Element	Description
MINUTES MIN	Optional keyword indicating that the preceding value is expressed in minutes.
SET <i>jobVariable</i> = ' <i>value</i> '	Optional job script specification to set any number of job variables and their default values. For more information, see the <i>Teradata® Parallel Transporter User Guide</i> , B035-2445.
STEP <i>stepName</i>	Required if there are multiple steps. Keyword phrase that names and introduces a job step. The <i>stepName</i> must be unique within the set of names of the steps in the job.
USING CHARACTER SET or USING CHAR SET <i>charsetId</i>	Optional keyword phrase preceding the DEFINE JOB statement that specifies the character set to be used for the job. This specification is needed only if you want to use an extended character set rather than the platform defaults of ASCII or EBCDIC. The <i>charsetId</i> is the name or code of the extended character set to be used for the job.

## Job Types

There are two types of jobs that can be defined with a DEFINE JOB statement:

- Jobs with a single APPLY statement not enclosed in the job STEP syntax, including jobs defined before Teradata PT supported multiple job steps.
- Jobs with one or more APPLY statements, each enclosed in the job STEP syntax.

## Example – Job with Single APPLY Statement

The following example shows a job with a single APPLY statement without using the job step syntax:

```

DEFINE JOB LOADPROD
DESCRIPTION 'LOAD PRODUCT DEFINITION TABLE'
(
  DEFINE SCHEMA PRODUCT_SOURCE_SCHEMA
  DESCRIPTION 'PRODUCT INFORMATION'
  (
    PRODUCT_NAME      VARCHAR(24),
    PRODUCT_CODE      INTEGER,
    PRODUCT_DESCRIPTION VARCHAR(512),
    PRODUCT_COST      INTEGER,
    PRODUCT_PRICE     INTEGER
  );
  DEFINE OPERATOR LOAD_OPERATOR

```

```

DESCRIPTION 'TERADATA PARALLEL TRANSPORTER LOAD OPERATOR'
TYPE LOAD
SCHEMA *
ATTRIBUTES
(
  INTEGER TenacityHours      = 0,
  INTEGER TenacitySleep       = 0,
  INTEGER BufferSize          = 16,
  INTEGER MaxSessions         = 1,
  INTEGER MinSessions         = 1,
  INTEGER ErrorLimit          = 1,
  VARCHAR TdpId               = 'MYDATABASE',
  VARCHAR UserName             = 'MYUSER',
  VARCHAR UserPassword         = 'MYPASSWORD',
  VARCHAR AccountId           = 'MYACCT',
  VARCHAR WorkingDatabase      = 'SALES',
  VARCHAR TargetTable          = 'SALES_TABLE',
  VARCHAR LogTable              = 'SALES.SALES_TABLE_LOG',
  VARCHAR ErrorTable1          = 'SALES.SALES_TABLE_ERROR1',
  VARCHAR ErrorTable2          = 'SALES.SALES_TABLE_ERROR2'
);

DEFINE OPERATOR DATACONN
DESCRIPTION 'TERADATA PARALLEL TRANSPORTER DATACONNECTOR OPERATOR'
TYPE DATACONNECTOR PRODUCER
SCHEMA PRODUCT_SOURCE_SCHEMA
ATTRIBUTES
(
  VARCHAR FileName            = 'sales_data.txt',
  VARCHAR OpenMode             = 'Read',
  VARCHAR Format               = 'FORMATTED',
  VARCHAR IndicatorMode        = ''
);

APPLY ('INSERT INTO SALES_TABLE (:PRODUCT_NAME,
                                :PRODUCT_CODE,
                                :PRODUCT_DESCRIPTION,
                                :PRODUCT_COST,
                                :PRODUCT_PRICE);')
TO OPERATOR (LOAD_OPERATOR [3])

SELECT * FROM (OPERATOR DATACONN);
);

```

## Job Steps

This feature allows multiple job operations in a single script to be executed sequentially. For example, the first step can execute a DDL operator to create a target table. The second step can then execute a Load operator to load the target table. This syntax is as follows:

```
<twb-job-body> ::= <metadata definitions> { (<job step>)+ |  
<executable statement> }
```

where the following is true:

Syntax	Description
<metadata definitions>	Non-executable statements, for example: DEFINE <script object>
<executable statement>	The APPLY statement
<job step>	Job step consists of the following: STEP <step name> '(' <executable statement> '+')' '.'
<step name>	A user-defined script identifier

## Example – Job Steps

The following example shows a job with a single APPLY statement using job step syntax:

```
DEFINE JOB LOADPROD  
DESCRIPTION 'LOAD PRODUCT DEFINITION TABLE'  
(  
    DEFINE SCHEMA PRODUCT_SOURCE_SCHEMA  
    DESCRIPTION 'PRODUCT INFORMATION'  
    (  
        PRODUCT_NAME      VARCHAR(24),  
        PRODUCT_CODE      INTEGER,  
        PRODUCT_DESCRIPTION VARCHAR(512),  
        PRODUCT_COST      INTEGER,  
        PRODUCT_PRICE     INTEGER  
    );  
  
    DEFINE OPERATOR DDL_OPERATOR  
    TYPE DDL
```

```

ATTRIBUTES
(
  VARCHAR TdpId          = 'MYDATABASE',
  VARCHAR UserName        = 'MYUSER',
  VARCHAR UserPassword    = 'MYPASSWORD'
);

DEFINE OPERATOR LOAD_OPERATOR
DESCRIPTION 'TERADATA PARALLEL TRANSPORTER LOAD OPERATOR'
TYPE LOAD
SCHEMA *
ATTRIBUTES
(
  INTEGER TenacityHours   = 0,
  INTEGER TenacitySleep   = 0,
  INTEGER BufferSize      = 16,
  INTEGER MaxSessions     = 1,
  INTEGER MinSessions     = 1,
  INTEGER ErrorLimit      = 1,
  VARCHAR TdpId           = 'MYDATABASE',
  VARCHAR UserName         = 'MYUSER',
  VARCHAR UserPassword    = 'MYPASSWORD',
  VARCHAR AccountId       = 'MYACCT',
  VARCHAR WorkingDatabase = 'SALES',
  VARCHAR TargetTable      = 'SALES_TABLE',
  VARCHAR LogTable         = 'SALES.SALES_TABLE_LOG',
  VARCHAR ErrorTable1      = 'SALES.SALES_TABLE_ERROR1',
  VARCHAR ErrorTable2      = 'SALES.SALES_TABLE_ERROR2'
);

DEFINE OPERATOR DATACONN
DESCRIPTION 'TERADATA PARALLEL TRANSPORTER DATACONNECTOR OPERATOR'
TYPE DATACONNECTOR PRODUCER
SCHEMA PRODUCT_SOURCE_SCHEMA
ATTRIBUTES
(
  VARCHAR FileName         = 'sales_data.txt',
  VARCHAR OpenMode          = 'Read',
  VARCHAR Format            = 'FORMATTED',
  VARCHAR IndicatorMode
);
Step Setup_Tables
(
  APPLY

```

```

('DROP    TABLE SALES.SALES_TABLE_LOG;'),
('DROP    TABLE SALES.SALES_TABLE_ERROR1;'),
('DROP    TABLE SALES.SALES_TABLE_ERROR2;'),
('DROP    TABLE SALES.SALES_TABLE;'),
('CREATE TABLE SALES.SALES_TABLE (NAME          VARCHAR(24),
                                  CODE           INTEGER,
                                  DESCRIPTION   VARCHAR(512),
                                  COST           INTEGER,
                                  PRICE          INTEGER);')

TO OPERATOR (DDL_OPERATOR)
);

Step Load_Table
(
  APPLY ('INSERT INTO SALES_TABLE (:PRODUCT_NAME,
                                   :PRODUCT_CODE,
                                   :PRODUCT_DESCRIPTION,
                                   :PRODUCT_COST,
                                   :PRODUCT_PRICE);')
  TO OPERATOR (LOAD_OPERATOR [3])

  SELECT * FROM (OPERATOR DATACONN);
);
);

```

## DEFINE SCHEMA

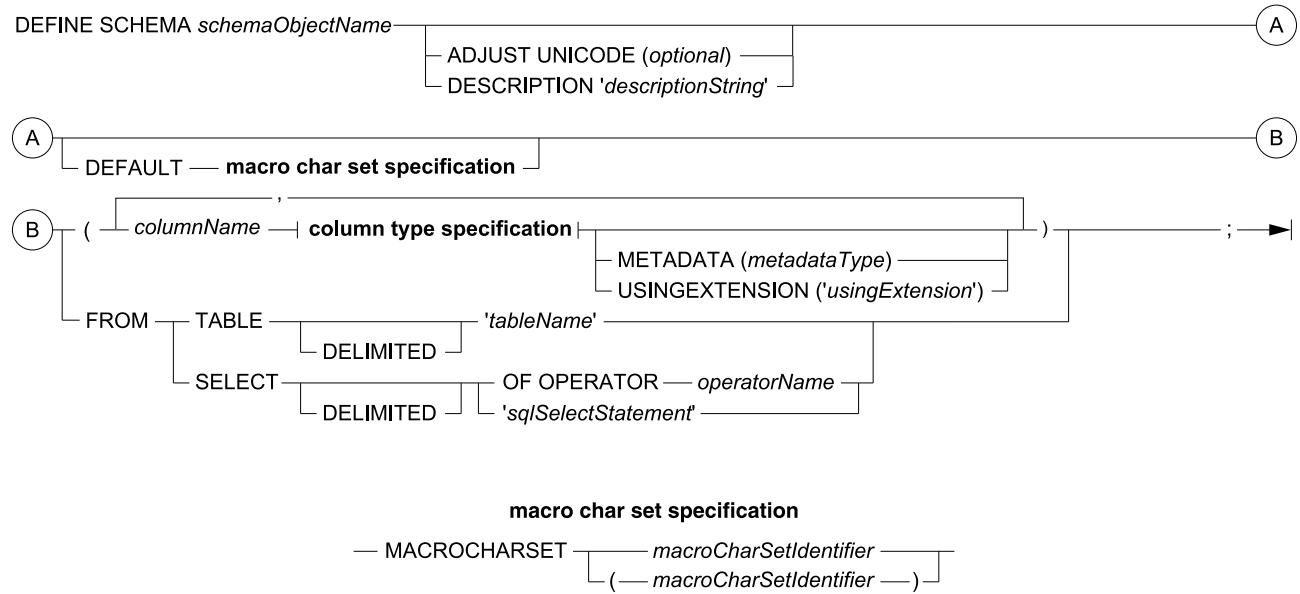
### Purpose

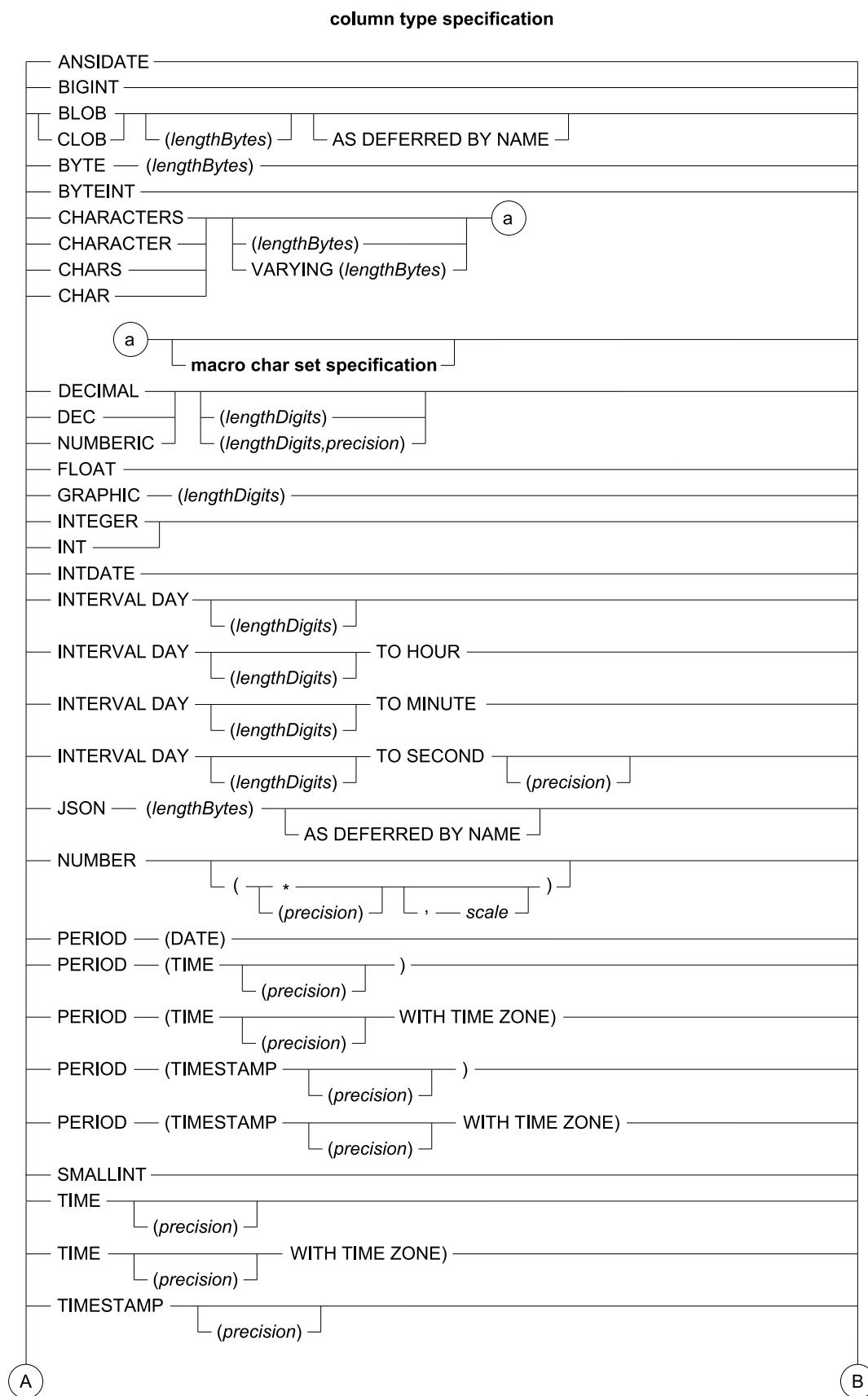
The DEFINE SCHEMA statement describes the structure of the data source or data target with an ordered set of column definitions. Each column definition consists of a column name and a Teradata PT data type identifier.

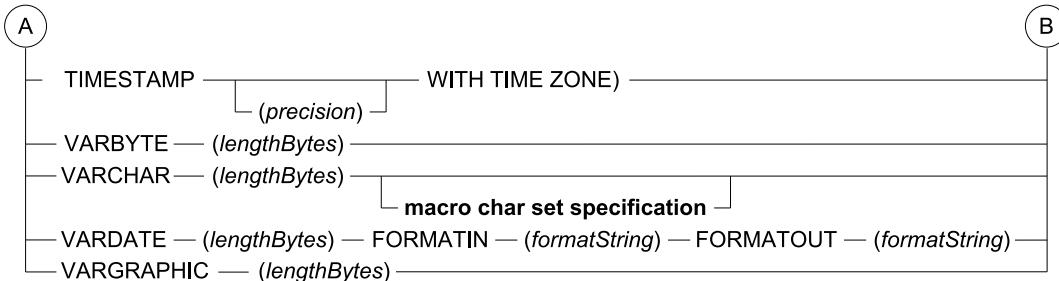
Some data types require additional defining attributes, such as lengthBytes for character data types and precision for data types with fractional seconds, such as TIME, TIMESTAMP, and some INTERVAL and PERIOD data types.

A job script can contain multiple schemas, and a schema can be used with multiple operators, if it accurately describes the data processed by each operator.

## Syntax





**column type specification (continued)**

where:

Syntax Element	Description
ANSIDATE	Optional keyword specifying a date data type in standard 10-character, dash-separated format. For example, October 23, 2007 is expressed as: 2007-10-23
BIGINT	Optional keyword specifying an 8-byte binary integer numeric data type.
BLOB( <i>lengthBytes</i> ) CLOB( <i>lengthBytes</i> )	<p>Optional keyword specifying an inline LOB (Large Object) column whose data is stored in the data rows defined by the schema just as non-LOB columns data is.</p> <p>An inline LOB is either of data type BLOB (Binary Large Object or CLOB (Character Large Object).</p> <p><i>lengthBytes</i> specifies the maximum number of bytes of data that an inline LOB column value can contain in a data row and like a VARCHAR column, it can be of any size smaller than <i>lengthBytes</i>. The largest inline LOB supported by Teradata PT is 2,097,088,000 bytes on non-z/OS platforms. The largest inline LOB supported by Teradata PT is 64,000 bytes on z/OS platform.</p> <p>The DataConnector operator reads or writes LOB data from or to an external file. The SQL Selector operator extracts LOB data from the database and puts it into data streams. The SQL Inserter, Stream, and Update operators can read data from the data stream and load it into the database.</p> <p>Teradata PT may transfer inline LOBs in deferred mode from the producer to the consumer if the inline LOBs cannot fit within a single request message. The user does not have to specify anything. TPT will automatically perform the transfer as needed. This feature is only valid on non-z/OS platforms.</p> <p><b>Note:</b></p> <p>An 8-byte length indicator precedes the inline LOB data if data rows are coming from a file read by a DataConnector operator. Other operators (Export and Load) cannot process LOB data.</p>
BLOB( <i>lengthBytes</i> ) AS DEFERRED BY NAME	Optional keyword specifying a deferred LOB (Large Object) column whose data is contained in files rather than in the data rows defined by a Teradata PT schema.

Syntax Element	Description
CLOB( <i>lengthBytes</i> ) AS DEFERRED BY NAME	<p>A deferred LOB is either of data type BLOB or CLOB. <i>lengthBytes</i> specifies the maximum number of bytes of data that a LOB column value can contain in the database and thus the maximum size of the files whose names are stored in the data rows. The maximum value of <i>lengthBytes</i> for deferred LOBs is 2,097,088,000.</p> <p>The names of the files that contain deferred LOB data are stored in data rows exactly like VARCHAR column values, with a 2-byte length prefix. The row value is either a fully qualified path name or the name of a file that Teradata PT looks for in the directory out of which it executes.</p> <p>The option AS DEFERRED BY NAME, which specifies a Teradata PT LOB file identifier, means that LOB data will be deferred and sent to the database separately from non-LOB data.</p> <p>The presence of the option AS DEFERRED BY NAME indicates that LOB file identifiers are expected in the data stream instead of the LOB data itself.</p> <p>The omission of AS DEFERRED BY NAME in the BLOB or CLOB column definition directs Teradata PT to send BLOB or CLOB data to the database in the data row, just like non-LOB data.</p> <p>The SQL Selector operator is the only producer operator that can extract LOB data from the database. The SQL Inserter, Update, and Stream operators are the only consumer operators that can load deferred LOBs into the database. Other operators (Export and Load) cannot process LOB data.</p> <p><b>Note:</b> Deferred LOB is not supported on z/OS platform.</p>
BYTE BYTE ( <i>lengthBytes</i> )	<p>Optional keyword specifying a fixed-length column containing binary data.</p> <p><i>lengthBytes</i> specifies the column length in bytes.</p> <p>The maximum valid length is 64,000 bytes. If not specified, the default length is 1 byte.</p>
BYTEINT	Optional keyword specifying a 1-byte integer, a whole number in the range -128 to +127.
CHAR CHARS CHARACTER CHARACTERS CHAR ( <i>lengthBytes</i> ) CHARS( <i>lengthBytes</i> ) CHARACTER( <i>lengthBytes</i> ) CHARACTERS( <i>lengthBytes</i> )	<p>Optional keyword that specifies a fixed-length column containing character data.</p> <p><i>lengthBytes</i> specifies the column length in bytes. The maximum valid length is 64,000. If not specified, the default length is 1.</p>
CHAR VARYING CHARS VARYING CHARACTER VARYING CHARACTERS VARYING	<p>Optional keyword specifying a variable-length column containing character data.</p> <p><i>lengthBytes</i> specifies the column length in bytes. The maximum valid length is 64,000. If not specified, the default length is 1.</p>

Syntax Element	Description
CHAR VARYING( <i>lengthBytes</i> ) CHARS VARYING( <i>lengthBytes</i> ) CHARACTER VARYING( <i>lengthBytes</i> ) CHARACTERS VARYING( <i>lengthBytes</i> )	See description of VARCHAR.
<i>columnName</i>	Required user-supplied name of a column in the data source or data target described by the schema. Each specified column name must be unique within the schema.
DECIMAL, DEC, or NUMERIC • ( <i>lengthDigits</i> ) • ( <i>lengthDigits</i> , <i>precision</i> )	<p>Optional keyword defining a decimal numeric data type. This data type also defines the length in digits and decimal precision for the data column.</p> <ul style="list-style-type: none"> <li>• <i>lengthDigits</i> is the total number of digits. The range of the <i>lengthDigits</i> specification is from 1 to 38. The default is 5.</li> <li>• The <i>lengthDigits</i> specification is required if a <i>precision</i> is specified.</li> <li>• <i>precision</i> is the number of digits to the right of the decimal point. The <i>precision</i> specification can range from 0 to the value of the <i>lengthDigits</i> specification. The default for precision is 0.</li> </ul> <p>If the <i>lengthDigits</i> and precision values are not specified, the default values are used.</p> <p>Mismatching DECIMAL data are allowed as long as the job adheres to these three rules:</p> <ol style="list-style-type: none"> <li>1. The number of digits on the source must be less than the number of digits in the schema.</li> <li>2. The storage size on the source and schema must be equal.</li> <li>3. The precision values must match.</li> </ol> <p>Since the database stores DECIMAL data as scaled integers, a different precision will result in incorrect values being stored in the database.</p>
DEFINE SCHEMA	Required keyword phrase specifying the beginning of the schema definition.
DELIMITED	Keyword indicating that Teradata PT is to generate the delimited-file format version of the Teradata PT schema based on one of the following: The table name specified: <ul style="list-style-type: none"> <li>• Via the quoted string <i>tableName</i>.</li> <li>• In <i>sq&gt;SelectStatement</i></li> <li>• By the <i>selectStmt</i> attribute of the script-defined operator.</li> </ul>
DESCRIPTION ' <i>descriptionString</i> '	Optional keyword phrase providing a descriptive comment about the defined schema.
FLOAT	Optional keyword defining the floating point data type.
FROM SELECT	Keyword phrase indicating that TPT is to generate a schema based on the columns in the result table of an SQL SELECT statement, specified either explicitly or indirectly via the SelectStmt attribute of operator <i>operatorName</i> .

Syntax Element	Description
FROM TABLE	Keyword phrase indicating that Teradata PT is to generate a schema based on the columns of the table identified via the quoted string <i>tableName</i> .
GRAPHIC GRAPHIC( <i>lengthChars</i> )	<p>Optional keyword specifying a fixed-length column containing 2-byte graphic characters.</p> <p><i>lengthChars</i> specifies the column length in 2-byte graphic characters. The maximum valid length is 32,000 characters. If not specified, the default length is 1 character.</p> <p>The byte length of a GRAPHIC column is twice the value of <i>lengthChars</i>.</p>
INTDATE	<p>Optional keyword specifying a column containing data values represented by 4-byte binary integers, calculated as follows:</p> $\text{intdate value} = (\text{year} - 1900) * 10000 + \text{month} * 100 + \text{day}$ <p>For example, the INTDATE column value for october 23, 2007 is 1071023</p>
INTEGER INT	Optional keyword specifying a four-byte integer data type.
INTERVAL DAY INTERVAL DAY( <i>lengthDigits</i> )	<p>Optional keyword specifying a time interval column in days, where <i>lengthDigits</i> is the number of digits used to express the interval, populated from right to left. Valid values are 1 through 4. The default value is 2.</p> <p><b>Note:</b>            External to the database, the values of INTERVAL columns are represented by character strings of various lengths and formats. Teradata PT, however, currently has no knowledge of these valid INTERVAL lengths and formats, and to Teradata PT, INTERVAL column values are indistinguishable from character strings in general. A job script can contain a character string that is intended to represent the value of an INTERVAL column, but Teradata PT cannot verify its validity. Assuming an INTERVAL column literal string is correctly formatted, it can reliably be compared to the corresponding type of INTERVAL column value in a data row only using either the "equals" (=) or the "not equals" (&lt;&gt;) comparison operators. For details on the valid lengths and formats of string representations of the values of INTERVAL column types, see <i>Teradata Vantage™ - Data Types and Literals</i>, B035-1143.</p>
INTERVAL DAY TO HOUR INTERVAL DAY( <i>lengthDigits</i> ) TO HOUR	<p>Optional keyword specifying a time interval column in days and hours stated in the named column, where <i>lengthDigits</i> is the number of digits used to define the interval in days, populated from right to left.</p> <p>Valid values are 1 through 4. The default value is 2.            See Note for INTERVAL DAY.</p>
INTERVAL DAY TO MINUTE INTERVAL DAY( <i>lengthDigits</i> ) TO MINUTE	Optional keyword specifying a time interval column in days, hours, and minutes for the named column, where <i>lengthDigits</i> is the number of digits used to express the interval in days, populated from right to left.

Syntax Element	Description
	Valid values are 1 through 4. The default value is 2. See Note for INTERVAL DAY.
INTERVAL DAY TO SECOND INTERVAL DAY( <i>lengthDigits</i> ) TO SECOND INTERVAL DAY TO SECOND( <i>precision</i> ) INTERVAL DAY( <i>lengthDigits</i> ) TO SECOND( <i>precision</i> )	Optional keyword specifying a time interval column in days, hours, minutes, and seconds where the following is true: <ul style="list-style-type: none"><li>• <i>lengthDigits</i> is the number of digits used to define the interval in days, populated from right to left. Valid values are 1 through 4. The default value is 2.</li><li>• <i>precision</i> is the number of digits to the right of the decimal point used to define fractions of a second. Valid values are 1 through 6. The default value is 6.</li></ul> See Note for INTERVAL DAY.
INTERVAL HOUR INTERVAL HOUR( <i>lengthDigits</i> )	Optional keyword specifying a time interval column in hours, where <i>lengthDigits</i> is the number of digits used to express the hour interval, populated from right to left. Valid values are 1 through 4, with a default value of 2. See Note for INTERVAL DAY.
INTERVAL HOUR TO MINUTE INTERVAL HOUR( <i>lengthDigits</i> ) TO MINUTE	Optional keyword specifying a time interval column in hours and minutes, where <i>lengthDigits</i> is the number of digits used to express the hour interval, populated from right to left. Valid values are 1 through 4, with a default value of 2. See Note for INTERVAL DAY.
INTERVAL HOUR TO SECOND INTERVAL HOUR( <i>lengthDigits</i> ) TO SECOND INTERVAL HOUR TO SECOND( <i>precision</i> ) INTERVAL HOUR( <i>lengthDigits</i> ) TO SECOND ( <i>precision</i> )	Optional keyword specifying a time interval column in hours, minutes, and seconds where: <ul style="list-style-type: none"><li>• <i>lengthDigits</i> is the number of digits used to define the interval hours, populated from right to left. Valid values are 1 through 4. The default value is 2.</li><li>• <i>precision</i> is the precision (number of digits to the right of the decimal point) used to define fractions of a second. Valid values are 1 through 6. The default value is 6.</li></ul> See Note for INTERVAL DAY.
INTERVAL MINUTE INTERVAL MINUTE( <i>lengthDigits</i> )	Optional keyword specifying a time interval column in minutes, where <i>lengthDigits</i> is the number of digits used to express the minute interval, populated from right to left. Valid values are 1 through 4, with a default value of 2. See Note for INTERVAL DAY.
INTERVAL MINUTE TO SECOND INTERVAL MINUTE( <i>lengthDigits</i> ) TO SECOND ( <i>precision</i> ) INTERVAL MINUTE( <i>lengthDigits</i> ) TO SECOND INTERVAL MINUTE TO SECOND( <i>precision</i> )	Optional keyword specifying a time interval column in minutes and seconds where: <ul style="list-style-type: none"><li>• <i>lengthDigits</i> is the number of digits used to define the interval in minutes, populated from right to left. Valid values are 1 through 4. The default value is 2.</li><li>• <i>precision</i> is the precision (number of digits to the right of the decimal point) used to define fractions of a second. Valid values are 1 through 6. The default value is 6.</li></ul> See Note for INTERVAL DAY.

Syntax Element	Description
INTERVAL SECOND, INTERVAL SECOND( <i>lengthDigits</i> ) INTERVAL SECOND( <i>lengthDigits</i> , <i>precision</i> )	<p>Optional keyword specifying a time interval column in seconds, optionally to the nearest fraction of a second, where:</p> <ul style="list-style-type: none"> <li>• <i>lengthDigits</i> is the number of digits used to define the interval in seconds, populated from right to left. Valid values are 1 through 4. The default value is 2.</li> <li>• <i>precision</i> is the precision (number of digits to the right of the decimal point) used to define fractions of a second. Valid values are 1 through 6. The default value is 6.</li> </ul> <p>See Note for INTERVAL DAY.</p>
INTERVAL MONTH INTERVAL MONTH( <i>lengthDigits</i> )	<p>Optional keyword specifying a time interval in months, where <i>lengthDigits</i> is the number of digits used to express the months, populated from right to left.</p> <p>Valid values are 1 through 4 with a default value of 2.</p> <p>See Note for INTERVAL DAY.</p>
INTERVAL YEAR INTERVAL YEAR( <i>lengthDigits</i> )	<p>Optional keyword specifying a time interval column in years, where <i>lengthDigits</i> is the number of digits used to express the year, populated from right to left.</p> <p>Valid values are 1 through 4, with a default value of 2.</p> <p>See Note for INTERVAL DAY.</p>
INTERVAL YEAR TO MONTH INTERVAL YEAR( <i>lengthDigits</i> ) TO MONTH	<p>Optional keyword specifying a time interval column in years and months, where <i>lengthDigits</i> is the number of digits used to express the year, populated from right to left.</p> <p>Valid values are 1 through 4, with a default value of 2.</p> <p>See Note for INTERVAL DAY.</p>
JSON( <i>lengthBytes</i> ) JSON( <i>lengthBytes</i> ) AS DEFERRED BY NAME	<p>Optional keyword specifying an inline JavaScript Object Notation (JSON) column whose data is stored in the data rows defined by the schema just as non-JSON column data is stored.</p> <p><i>lengthBytes</i> specifies the maximum number of bytes of data that an in-line JSON column can contain in a data row and like a VARCHAR column, it can be any size smaller than <i>lengthBytes</i>. The largest inline JSON column supported by Teradata PT is 16,776,192 bytes on non-z/OS platforms. The largest inline JSON column supported by Teradata PT is 64,000 bytes on z/OS platform.</p> <p>The DataConnector operator reads and writes JSON data from and to an external file. The SQL Selector operator extracts JSON data from the database and puts it into data streams. The SQL Inserter, Update, and Stream operators read data from the data stream and load it into the database.</p> <p>Teradata PT may transfer inline JSON columns in deferred mode from the producer to the consumer if the inline JSON columns cannot fit within a single request message. The user does not have to specify anything. TPT will automatically perform the transfer as needed. This feature is only valid on non-z/OS platforms.</p>

Syntax Element	Description
	<p><b>Note:</b></p> <p>An 8-byte length indicator precedes the in-line JSON data if data rows are coming from a file read by the DataConnector operator. Other operators (Export and Load) cannot process JSON data.</p> <p><b>AS DEFERRED BY NAME</b></p> <p>Optional keyword specifying a deferred JSON column whose data is contained in files rather than in the data rows defined by a Teradata PT schema.</p> <p><i>lengthBytes</i> specifies the maximum number of bytes of data that a JSON column can contain in the database and thus the maximum size of the files whose names are stored in the data rows. The maximum value of <i>lengthBytes</i> for deferred JSON files is 16,776,192 bytes.</p> <p>The presence of the option AS DEFERRED BY NAME indicates that JSON file identifiers are expected in the data stream instead of the JSON data itself. The names of the files that contain deferred JSON data are stored in data rows exactly like VARCHAR column values, with a 2-byte length prefix. The row value is either a fully qualified path name or the name of a file that Teradata PT looks for in the directory out of which it executes.</p> <ul style="list-style-type: none"> <li>• The option AS DEFERRED BY NAME, which specifies a Teradata PT JSON file identifier, means that JSON data will be deferred and sent to the database separately from non-JSON data.</li> <li>• The omission of AS DEFERRED BY NAME in the JSON column definition directs Teradata PT to send JSON data to the database in the data row, just like non-JSON data.</li> </ul> <p>SQL Selector operator is the only producer operator that can extract JSON data from the database. The SQL Inserter operator is the only consumer operator that can load deferred JSON files into the database. Other operators (Export, Load, Update and Stream) cannot process JSON data.</p> <p><b>Note:</b></p> <p>Deferred JSON is not supported on z/OS platform.</p>
<i>lengthBytes</i>	<p>Length specification for nonnumeric data types.</p> <p><i>lengthBytes</i> specification is required for variable-length data types.</p> <p><b>Note:</b></p> <p>When <i>lengthBytes</i> is specified for a CHAR or VARCHAR column encoded in UTF-16, it represents the number of bytes occupied by the column values, which is twice the character count that would be specified for length in an equivalent column description of a table schema.</p>
<i>lengthChars</i>	<p>Optional keyword specifying the length in 2-byte graphic characters of the values of a GRAPHIC or VARGRAPHIC column.</p> <p>The byte length of these values is twice the value of <i>lengthChars</i>.</p>

Syntax Element	Description
MACROCHARSET	<p>Optional clause that defines the database server character set name for the character field.</p> <p>The MACROCHARSET is used to generate a CHARACTER SET &lt;ServerCharsetName&gt; clause for each field of character data type in the Stream Operator macro creation.</p> <p>If there is no MACROCHARSET clause for a field of character type, then the default will be one of the following:</p> <ul style="list-style-type: none"> <li>• No CHARACTER SET clause, generated in the macro creation, when the client session character set is one of the following: <ul style="list-style-type: none"> <li>◦ ASCII</li> <li>◦ EBCDIC</li> <li>◦ EBCDIC037_0E</li> <li>◦ EBCDIC277_0E</li> <li>◦ EBCDIC273_0E</li> <li>◦ LATIN1_0A</li> <li>◦ LATIN9_0A</li> <li>◦ LATIN1252_0A</li> <li>◦ LATIN1252_3A0</li> <li>◦ Simple (single-byte) site-defined client character sets whose names end in _0A or _0E</li> </ul> </li> <li>• CHARACTER SET UNICODE clause, generated in the macro creation, for all other client session character sets not mentioned in this list.</li> </ul> <p>The MACROCHARSET clause applies only to:</p> <ul style="list-style-type: none"> <li>• Character fields. An error is returned when the MACROCHARSET is specified for non-character type fields.</li> <li>• The Teradata PT Stream Operator. The MACROCHARSET clause is ignored for other operators.</li> </ul>
<i>macroCharSetIdentifier</i>	<p>Optional identifier that specifies the server storage character set name the Stream Operator uses to generate a CHARACTER SET clause for each character column when the Stream Operator creates the macros that are necessary for loading the data.</p> <p>The five possible valid values are as follows:</p> <ul style="list-style-type: none"> <li>• LATIN</li> <li>• UNICODE</li> <li>• KANJI1</li> <li>• KANJISJIS</li> <li>• GRAPHIC</li> </ul> <p>Values specified for this identifier are meaningful in a DEFINE SCHEMA statement only when the schema is employed by an instance of the Stream operator.</p>
METADATA ( <i>metadataType</i> )	<p>Optional keyword specifying that the value for this schema column is not included in the source data, but instead is metadata supplied directly by the operator itself. This feature can help with identifying the origin of the data once it has been loaded, and can also help locate invalid source records that cannot be loaded by the database.</p> <p>The following values can be used for "metadataType":</p>

Syntax Element	Description
	<ul style="list-style-type: none"> <li>'FileName': The current filename will be added to the source data. It must be defined as a CHAR or VARCHAR data type, and the size of this column must be large enough to hold the longest filename encountered.</li> <li>'FileRecordNumber': The file's record number will be added to the source data for each record processed. It must be defined as an INTEGER or BIGINT data type. The target table column must be large enough to handle the maximum record number encountered. The largest value allowed for INTEGER is: 2,147,483,647. If the record number exceeds this maximum value, 0 will be stored instead.</li> </ul> <p>Only one entry for each "metadataType" is allowed. However, a METADATA column can occur anywhere within the schema. This feature is only valid when used with the \$FILE_READER operator. All DataConnector format types are supported.</p> <p>See examples 6 and 7 in <a href="#">Examples of DEFINE SCHEMA Statements</a>.</p>
NUMBER NUMBER(*) NUMBER(*,scale) NUMBER(precision) NUMBER(precision,scale)	<p>Optional keyword specifying a column whose values represent numbers up to 38 digits of precision, optionally up to precision digits of scale with an optional exponent with the range of:</p> <ul style="list-style-type: none"> <li>E-130 to E+125, if scale is zero.</li> <li>E-scale to E+(125 - scale), if scale is greater than zero.</li> </ul> <p><b>Note:</b> Columns defined as NUMBER in the TPT schema object cannot be referenced in boolean predicates or numeric value expressions in the TPT APPLY statement, unless the predicate or expression is solely made up of NUMBER column references. Predicates or expressions involving NUMBER column references and non-NUMBER column references are not supported.</p>
PERIOD(DATE)	<p>Optional keyword specifying a column whose values represent a duration of time, consisting of a beginning and ending date value.</p> <p><b>Note:</b> The values of PERIOD columns are internally coded, both within the database and within Teradata PT. There is no character string representation of the value of any PERIOD column type in a job script, and two columns of the same PERIOD type cannot even be compared in a boolean predicate. For details on the internal structure of the values of PERIOD column types and how they are represented when retrieved through BTEQ, see <i>Teradata Vantage™ - Data Types and Literals</i>, B035-1143.</p>
PERIOD(TIME) PERIOD(TIME( <i>precision</i> ))	<p>Optional keyword specifying a column whose values represent a duration of time, consisting of a beginning and an ending time value. <i>precision</i> specifies the number of fractional seconds digits, with valued values of 0 through 6. The default value is 6.</p>

Syntax Element	Description
	See Note for PERIOD(DATE).
PERIOD (TIME WITH TIME ZONE) PERIOD(TIME( <i>precision</i> )WITH TIME ZONE)	Optional keyword specifying a column whose values represent a duration of time, consisting of a beginning and an ending time-with-time-zone value. <i>precision</i> specifies the number of fractional seconds digits, with valid values of 0 through 6. The default value is 6. See Note for PERIOD(DATE).
PERIOD(TIMESTAMP) PERIOD(TIMESTAMP( <i>precision</i> ))	Optional keyword specifying a column whose values represent a duration of time, consisting of a beginning and an ending timestamp value. <i>precision</i> specifies the number of fractional seconds digits, with valid values of 0 through 6. The default value is 6. See Note for PERIOD(DATE).
PERIOD(TIMESTAMP WITH TIME ZONE) PERIOD(TIMESTAMP( <i>precision</i> ) WITH TIME ZONE)	Optional keyword specifying a column whose values represent a duration of time, consisting of a beginning and an ending timestamp-with-time-zone value. <i>precision</i> specifies the number of fractional seconds digits, with valid values of 0 through 6. The default value is 6. See Note for PERIOD(DATE).
<i>lengthDigits</i>	Optional length value in digits for data types that contain decimal numbers. The range and default value for <i>lengthDigits</i> vary according to the data type to which they are applied. For details see the individual data type descriptions in this table.
OF OPERATOR	Keyword phrase indicating that Teradata PT is to generate a schema based on the SQL SELECT statement assigned to the SelectStmt attribute of the script-defined <i>operatorName</i> .
<i>operatorName</i>	The name of an operator, defined in a DEFINE OPERATOR statement, whose SQL SELECT statement is the basis for a Teradata PT-generated schema. If there are no operators defined in the script, then the <i>operatorName</i> can be the name of an operator template specified in the APPLY-SELECT statement.
<i>precision</i>	Optional value that defines the number of digits to the right of the decimal point. The range and default values for <i>precision</i> vary according to the data type to which they are applied. For details the individual data type descriptions in this table.
<i>schemaObjectName</i>	Required internal Teradata PT metadata name of the defined schema object.
SMALLINT	Optional keyword specifying a two-byte integer numeric data type.
sqlSelectStatement	An SQL SELECT statement whose result table columns are the basis for a Teradata PT-generated schema.

Syntax Element	Description
<i>tableName</i>	<p>The name of a database table whose column definitions Teradata PT uses as the basis for generating:</p> <ul style="list-style-type: none"> <li>• A DEFINE SCHEMA statement that will replace this abbreviated version of the DEFINE SCHEMA statement in the job script</li> <li>• An SQL INSERT statement that will replace the immediately-preceding script macro \$INSERT in the job script.</li> </ul>
TIME TIME( <i>precision</i> )	<p>Optional keyword specifying a column whose values are time values.</p> <p><i>precision</i> represents the number of digits to the right of the decimal point, indicating a decimal fraction of a second. Valid values for <i>precision</i> are 0 through 6. The default value is 6.</p> <p><b>Note:</b></p> <p>In the 4 time-oriented column types - TIME, TIME WITH TIME ZONE, TIMESTAMP and TIMESTAMP WITH TIME ZONE - the default precision for the fractional seconds is 6, not zero. Thus TIME and TIME(6) represent identical columns, and you must specify TIME(0) if the time values will have no fractional seconds.</p> <p>External to the database, the values of these 4 time-oriented columns are represented by character strings of well-defined lengths and formats. Teradata PT, however, has no knowledge of these lengths or formats, and cannot verify the validity of job script character strings intended to represent values of these time-oriented columns.</p> <p>If a script literal string is a valid representation of the value of one of these column types, then all the comparison operators will correctly compare the value to row values of a column of exactly the same time-oriented type. Similarly, all comparison operators will correctly compare the row values of two columns of the same type.</p>
TIME WITH TIME ZONE TIME( <i>precision</i> ) WITH TIME ZONE	<p>Optional keyword specifying a column containing time values that include a time zone offset.</p> <p><i>precision</i> represents the number of digits to the right of the decimal point; a decimal fraction of a second. Valid values for <i>precision</i> are 0 through 6. The default value is 6.</p> <p>See Note for TIME.</p>
USINGEXTENSION (‘usingExtension’)	<p>Optional keyword specifying the USING clause extension for this schema column.</p> <p>For data loading, the operator generates the USING clause based on the schema. The operator sends the USING clause to the database. The USING clause identifies the layout of the data rows. The operator adds the UsingExtension value to the column when generating the USING clause. The operator does not validate the UsingExtension value. The UsingExtension value will be validated by the database.</p> <p>The USINGEXTENSION option applies only to the Teradata PT Update operator when the job is using the Extended MultiLoad Protocol.</p>

Syntax Element	Description
	<p>The USINGEXTENSION option is ignored when the job is using the traditional MultiLoad protocol. Also, the USINGEXTENSION option is ignored for other Teradata PT operators.</p> <p>The USINGEXTENSION option can be used to load data into a temporal table using the Teradata PT Update operator.</p> <p>For more information on how to use the USINGEXTENSION option, see <a href="#">Temporal Tables</a>.</p>
VARBYTE VARBYTE( <i>lengthBytes</i> )	<p>Optional keyword specifying a variable-length column containing binary data.</p> <p><i>lengthBytes</i> specifies the maximum column length in bytes.</p> <p>If not specified, the default column length is 64,000 bytes, which is the maximum valid length.</p> <p>VARCHAR and VARBYTE data with a length less than or equal to the defined length in the DEFINE SCHEMA definition is allowed.</p>
VARCHAR VARCHAR( <i>lengthBytes</i> )	<p>Optional keyword specifying a variable-length column containing character data.</p> <p><i>lengthBytes</i> specifies the maximum column length in bytes.</p> <p>If not specified, the default column length is 64,000 bytes, which is the maximum valid length.</p>
VARDATE( <i>lengthBytes</i> ) FORMATIN( <i>formatString</i> ) FORMATOUT( <i>formatString</i> )	<p>Optional keyword specifying a varying length character-based column whose data is to be reformatted before being loaded into a DateTime column.</p> <p>The FORMATIN string matches the format of the incoming DateTime data.</p> <p>The FORMATOUT string matches the format accepted by the DateTime column into which data is being loaded.</p> <p><i>lengthBytes</i> must be greater than or equal to the size of the larger format string.</p> <p>See “Using VARDATE Columns to Reformat DateTime Data” in <i>Teradata® Parallel Transporter User Guide</i>, B035-2445 for information on, and examples of, reformatting DateTime data using the VARDATE column data type.</p>
VARGRAPHIC VARGRAPHIC( <i>lengthBytes</i> )	<p>Optional keyword specifying a variable-length column containing 2-byte graphic characters.</p> <p><i>lengthBytes</i> specifies the column length in 2-byte graphic characters.</p> <p>If not specified, the default length is 32,000 characters, which is the maximum valid length.</p> <p>The byte length of VARGRAPHIC column is twice the value of <i>lengthChars</i>.</p>
XML XML AS DEFERRED BY NAME	<p>Optional keyword specifying an inline XML column whose data is stored in the data rows defined by the schema just as non-XML columns data is.</p> <p>Teradata PT defaults to size 2,097,088,000 bytes for inline XML columns on non-z/OS platforms. Teradata PT defaults to size 64,000 bytes for inline XML columns on z/OS platform.</p>

Syntax Element	Description
	<p>The DataConnector operator reads or writes XML data from or to an external file. The SQL Selector operator extracts XML data from the database and puts it into data streams. The SQL Inserter, Update, and Stream operators read data from the data stream and load it into the database.</p> <p>Teradata PT may transfer inline XML columns in deferred mode from the producer to the consumer if the inline XML columns cannot fit within a single request message. The user does not have to specify anything. TPT will automatically perform the transfer as needed. This feature is only valid on non-z/OS platforms.</p> <p><b>Note:</b></p> <p>An 8-byte length indicator precedes the inline XML data if data rows are coming from a file read by a DataConnector operator. Other operators (Export and Load) cannot process XML data.</p> <p><b>AS DEFERRED BY NAME</b></p> <p>Option specifying a deferred XML column whose data is contained in files rather than in the data rows defined by a Teradata PT schema.</p> <p>Teradata PT defaults to size 2,097,088,000 bytes for deferred XML columns.</p> <p>The option AS DEFERRED BY NAME, which specifies a Teradata PT XML file identifier, means that XML data is deferred and sent to the database separately from non-XML data.</p> <p>The names of the files that contain deferred XML data are stored in data rows exactly like VARCHAR column values, with a 2-byte length prefix. The row value is either a fully qualified path name or the name of a file that Teradata PT looks for in the directory out of which it executes.</p> <ul style="list-style-type: none"> <li>• The presence of the option AS DEFERRED BY NAME indicates that XML file identifiers are expected in the data stream instead of the XML data itself.</li> <li>• The omission of AS DEFERRED BY NAME in the XML column definition directs Teradata PT to send XML data to the database in the data row, just like non-XML data.</li> </ul> <p>SQL Selector operator is the only producer operator that can extract XML data from the database. The SQL Inserter, Update, and Stream operators can read XML data from the data stream and load it into the database. Other operators (Export and Load) cannot process XML data.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>• Teradata PT does not allow an XML column to be used in any predicate evaluation specified in the WHERE clause or in either type of CASE expression of an SQL request.</li> <li>• Deferred XML is not supported on z/OS platform.</li> </ul>

## Usage Notes

The following table describes the things to consider when using the DEFINE SCHEMA statement.

Topic	Description
Use of UNICODE Affects Column Width Requirements	<p>Teradata PT can automatically adjust the column widths when explicitly using UTF-8 and UTF-16 character sets with the ADJUST UNICODE keywords. See syntax diagram in <a href="#">DEFINE SCHEMA</a> for correct syntax of the keyword. If the USING CHARACTER SET UTF8 or UTF16 specifier and the ADJUST UNICODE specifier are both present in the script, column widths will automatically adjusted for Unicode type used.</p> <p><b>Note:</b></p> <p>Affected Datatypes are: CHAR, VARCHAR, CLOB and JSON.</p> <p>For more information on USING CHARACTER SET specifier details, see <a href="#">Using Extended Character Sets</a>.</p> <p>Alternatively, you may adjust the column widths manually. The rules for adjustments are as follows:</p> <ul style="list-style-type: none"> <li>When specifying a UTF-8 character set in a Teradata PT script the output schema definition must define column widths three times larger.</li> <li>When using the UTF-16 character set in a Teradata PT job script, the output SCHEMA definition must define column widths two times larger. The width values must be an even and positive number.</li> </ul>
Using Default Lengths	<p>Many data types are associated with length specifications <i>lengthBytes</i> or <i>lengthDigits</i>, which may be either optional or required, depending on the data type. Most of these allow for deferral to the default length value, by simply not specifying the length. However, since the default length is often the maximum length, use of the default value could cause a minor degradation in performance. Be sure to specify length values rather than deferring to the default values when maximizing performance is important.</p>
Specifying ANSI/SQL INTERVAL, TIME, and TIMESTAMP Datatypes	<p>In prior releases, Teradata PT required users to define columns of these types as CHAR columns of the appropriate lengths in Teradata PT schemas, for example, CHAR(8) for a Database TIME(0) column.</p> <p>Starting with Teradata PT Release 12.0, these columns can be defined with their database data type identifiers directly in DEFINE SCHEMA statements.</p>
Specifying ARRAY Data Types	<p>A column that is defined as an ARRAY data type in a database table must be specified as a VARCHAR data type in the DEFINE SCHEMA statement. The external representation for an ARRAY data type is VARCHAR.</p> <p>For more information, see the <i>Teradata® Parallel Transporter User Guide</i>, B035-2445.</p>
Specifying USINGEXTENSION	<p>The USINGEXTENSION option can be used to load data into a temporal table using the Teradata PT Update operator.</p> <p>For more information on how to use the USINGEXTENSION option, see <a href="#">Temporal Tables</a>.</p>
Mismatched Schemas	<p>A mismatched schema can yield unexpected results.</p>

## Examples of DEFINE SCHEMA Statements

The following table lists examples of DEFINE SCHEMA statements.

**Examples of DEFINE SCHEMA Statements**

Example	Statement Example
Example 1	<pre>DEFINE SCHEMA PRODUCT_SOURCE_SCHEMA DESCRIPTION 'PRODUCT INFORMATION' (     PRODUCT_NAME          VARCHAR(24),     PRODUCT_CODE          INTEGER,     PRODUCT_DESCRIPTION   VARCHAR(512),     PRODUCT_COST          INTEGER,     PRODUCT_PRICE         INTEGER,     PRODUCT_WARRANTY     CLOB(50000),     PRODUCT_PICTURE       BLOB (40000000) AS DEFERRED BY NAME );</pre>
Example 2	<pre>DEFINE SCHEMA FREQ_FLYER_SRC ADJUST UNICODE DESCRIPTION 'SCHEMA FOR FREQUENT FLYER SOURCE FILE' (     RES_ID VARCHAR(14),     FLIGHT_NO SMALLINT,     ORIGIN CHAR(3),     DEST CHAR(3),     TKT_ID CHAR(12),     TKT_ISSUE_DATE ANSIDATE,     TKT_AMT DECIMAL(5,2),     TRAVEL_AGCY_ID CHAR(4),     FREQ_MILES_ID CHAR(8) );</pre>
Example 3	<pre>DEFINE SCHEMA PRODUCT_SOURCE_SCHEMA DESCRIPTION 'PRODUCT INFORMATION' (     PRODUCT_CODE          INTEGER,     PRODUCT_NAME          VARCHAR(24)  MACROCHARSET UNICODE,     PRODUCT_DESCRIPTION   VARCHAR(512) MACROCHARSET LATIN,     PRODUCT_TYPE          CHAR(2)      MACROCHARSET KANJI,     PRODUCT_COST          INTEGER,     PRODUCT_PRICE         INTEGER );</pre>
Example 4	<pre>DEFINE SCHEMA EMPLOYEE_SCHEMA DESCRIPTION 'SAMPLE EMPLOYEE SCHEMA' (     EMP_ID    INTEGER,     EMP_NAME   CHAR(30),     EMP_DEPT   INTEGER,</pre>

Example	Statement Example
	<pre data-bbox="311 291 1209 361">JOB DURATION PERIOD(DATE) USING EXTENSION('AS VALIDTIME') );</pre>
Example 5	<pre data-bbox="355 418 1274 734">DEFINE SCHEMA EMPLOYEE_SCHEMA DESCRIPTION 'SAMPLE EMPLOYEE SCHEMA' (     EMP_ID      INTEGER,     EMP_NAME    CHAR(30),     EMP_DEPT    INTEGER,     JOB_START   INTDATE,     JOB_END     INTDATE USING EXTENSION('PERIOD FOR JOBDURATION                                 (JOB_START, JOB_END) AS VALIDTIME') );</pre>
Example 6	<pre data-bbox="355 813 1090 1108">DEFINE SCHEMA INVENTORY_SCHEMA1 DESCRIPTION 'SAMPLE INVENTORY SCHEMA' (     FILE_NAME   VARCHAR(30) METADATA(FILENAME),     REC_NUMBER  BIGINT METADATA(FILERECORDNUMBER),     ITEM_NUMBER INTEGER,     DESCRIPTION  VARCHAR(100),     QUANTITY    INTEGER,     PRICE       DECIMAL(6,2) );</pre>
Example 7	<pre data-bbox="355 1172 1106 1467">DEFINE SCHEMA INVENTORY_SCHEMA2 DESCRIPTION 'SAMPLE INVENTORY SCHEMA' (     ITEM_NUMBER  INTEGER,     DESCRIPTION   VARCHAR(100),     QUANTITY     INTEGER,     PRICE        DECIMAL(6,2),     REC_NUMBER   INTEGER METADATA(FILERECORDNUMBER),     FILE_NAME    CHAR(25) METADATA(FILENAME) );</pre>

## DEFINE OPERATOR

### Purpose

The DEFINE OPERATOR statement defines a Teradata PT operator object. Operator objects identify the Teradata PT operators, that is, the job components that actually perform the movement of data from sources to targets, as well as other supporting job functions.

Teradata PT provides a set of standard (or predefined) operators. These operators can be identified by means of the TYPE keyword as follows:

TYPE	Definition
DATACONNECTOR PRODUCER	DataConnector operator used as a producer
DATACONNECTOR CONSUMER	DataConnector operator used as a consumer
DDL	DDL operator
EXPORT	Export operator
FASTEXPORT OUTMOD	FastExport OUTMOD Adapter operator
FASTLOAD INMOD	FastLoad INMOD Adapter operator
INSERTER	SQL Inserter operator
LOAD	Load operator
MULTILOAD INMOD	MultiLoad INMOD Adapter operator
ODBC	ODBC operator
OS COMMAND	OS Command operator
SCHEMAMAPPER	Schema Mapping operator
SELECTOR	SQL Selector operator
STREAM	Stream operator
UPDATE	Update operator

For details on the capabilities, attributes, and required syntax for each of these standard Teradata PT operators, see the corresponding topics in this document.

Teradata PT allows you to define operators using a more generic use of the TYPE keyword. For example, those who wish to create their own operators must define TYPE using the following TYPE specifications:

TYPE	Definition
PRODUCER	Operators that produce a data stream from data received from a data source.
CONSUMER	Operators that consume data from a data stream and load it into a data target.
FILTER	Operators that filter data moving between a data source and a data target.
STANDALONE	Operators that work alone without receiving data from, or sending data to, other operators.

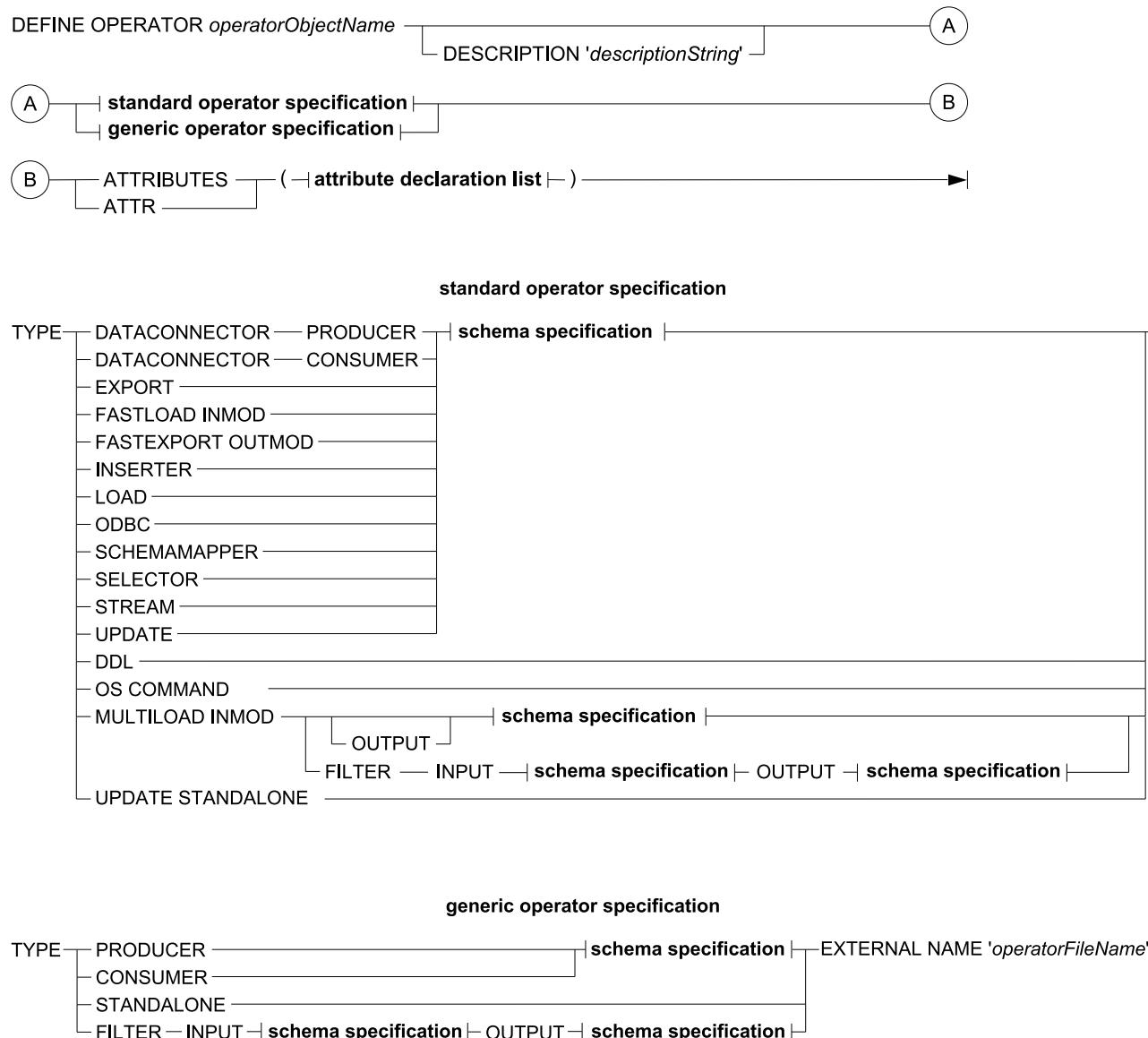
For details on creating customer-coded operators *Teradata Parallel Transporter Operator Programmer Guide* (B035-2516).

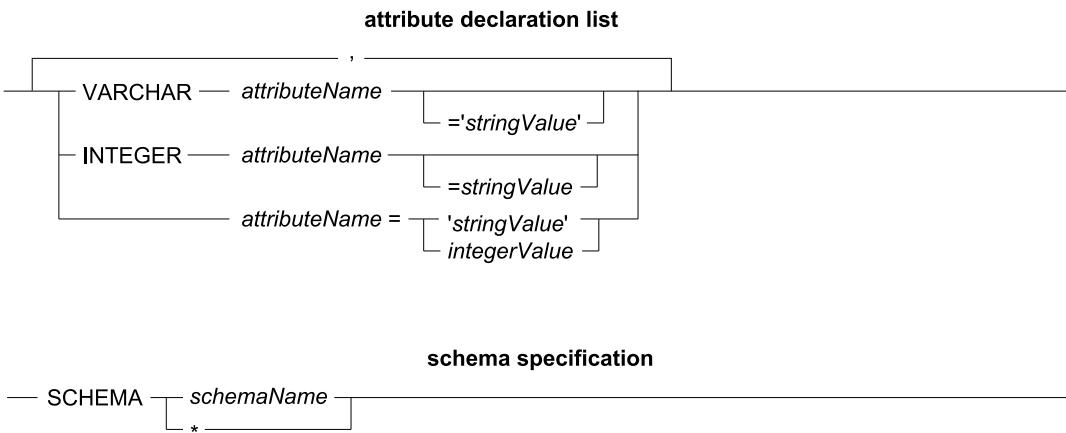
Teradata PT has internal knowledge of its standard operators, such as the operator library file and its message catalog file, so that these properties need not be specified in the DEFINE OPERATOR statement.

Teradata PT has no such knowledge about a customer-coded operator, so the former must be specified via the EXTERNAL NAME keyword phrase and the latter via the MSGCATALOG attribute, if a standard name is not used. See MSGCATALOG “catalog-name” in [Deprecated Syntax](#).

## Syntax

The DEFINE OPERATOR syntax order is important.





where:

Syntax Element	Description
<i>attributeName</i>	The name of an attribute that is meaningful to the operator being defined. For example, the value of the attribute name supplies the operator with required information, invokes an optional operator feature, or otherwise helps to direct the activities of the operator during job execution.
ATTRIBUTES ATTR	Required keyword introducing a list of attribute declarations for the defined operator object. For information on available attributes and required syntax see the topics describing the individual operators, beginning with <a href="#">DataConnector Operator</a> .
attribute declaration list	Required. Excerpt representing a list of the attribute declarations for the operator object. Observe the following when declaring attributes: <ul style="list-style-type: none"> <li>• All predefined Teradata PT operators have some required attributes, all of which must be declared.</li> <li>• Optional attributes for predefined Teradata PT operators automatically use default attribute values unless alternate values are specified.</li> <li>• Declare optional attributes only if you want to assign values to those attribute that do not have defaults, or to change the default values, either in the operator definition, or in a later section of the job script, such as an APPLY statement.</li> </ul> For detailed strategy on how to determine the most useful locations for assigning attribute values “Declaration of Operator Attributes” in <i>Teradata Parallel Transporter User Guide</i> (B035-2445). For the attribute descriptions and syntax the topics describing the individual operators, beginning with <a href="#">DataConnector Operator</a> .
DEFINE OPERATOR	Required keyword phrase specifying the beginning of the operator definition.
DESCRIPTION ' <i>descriptionString</i> '	Optional keyword phrase providing a descriptive comment about the defined operator object.
EXTERNAL NAME ' <i>operatorFileName</i> '	Keyword phrase indicating the actual name of a generic or custom operator library file.

Syntax Element	Description
	This attribute is required if specifying a generic or custom operator type (for example, TYPE CONSUMER), and not required if specifying a standard, predefined operator type (for example, TYPE STREAM).
<i>integerValue</i>	An integer that is a valid value of the operator attribute being declared: INTEGER type attribute <i>attributeName</i> .
<i>operatorObjectName</i>	Required internal Teradata PT metadata name of the defined operator object.
standard operator specification	<p>Excerpt representing a list of the standard operator type specifications for the TYPE specification.</p> <p>Specify one of the following:</p> <ul style="list-style-type: none"> <li>• DATACONNECTOR PRODUCER</li> <li>• DATACONNECTOR CONSUMER</li> <li>• DDL</li> <li>• EXPORT</li> <li>• FASTEXPORT OUTMOD</li> <li>• FASTLOAD INMOD</li> <li>• INSERTER</li> <li>• LOAD</li> <li>• MULTILOAD INMOD</li> <li>• MULTILOAD INMOD FILTER</li> <li>• ODBC</li> <li>• OS COMMAND</li> <li>• SCHEMAMAPPER</li> <li>• SELECTOR</li> <li>• STREAM</li> <li>• UPDATE</li> <li>• UPDATE STANDALONE</li> </ul>
<i>stringValue</i>	A character string that is a valid value of the operator attribute being declared: VARCHAR type attribute <i>attributeName</i> .
generic operator specification	<p>Excerpt representing a list of the generic operator type specifications for the TYPE specification.</p> <p>Specify one of the following depending on how the operator functions:</p> <ul style="list-style-type: none"> <li>• <b>Producer</b>: specifies that the defined operator object is of type producer, meaning that it may retrieve data from an external data store such as a file or database table, and provides it to other operators.</li> <li>• <b>Consumer</b>: specifies that the defined operator object is of type consumer, meaning that it accepts data from other operators and may store it in an external data store such as a file or database table.</li> <li>• <b>Standalone</b>: specifies that the defined operator does not exchange data with other operators.</li> <li>• <b>Filter</b>: specifies that the defined operator object will perform filtering on data en route from other operators.</li> </ul> <p>If one of these generic custom operator types is specified, then EXTERNAL NAME is also required.</p>

Syntax Element	Description
SCHEMA <i>schemaName</i> SCHEMA *	<p>Keyword phrase identifying a Teradata PT schema, previously defined in the job script, or *, deferring schema identification until run time.</p> <p>Observe the following operator schema requirements:</p> <ul style="list-style-type: none"> <li>• Producer operators must specify a schema.</li> <li>• Consumer operators must specify SCHEMA* (“deferred” schema), meaning that the consumer operator uses the schema that the producer operator uses. The schema is passed to the consumer operator during job execution.</li> <li>• Standalone operators must not specify a schema.</li> <li>• Filter operators must specify both an input and an output schema.</li> </ul>
TYPE	<p>Required keyword specifying the type of operator object to be used in the job. Using the form TYPE &lt;operator specification&gt; specify a standard, predefined operator, or a generic operator.</p> <ul style="list-style-type: none"> <li>• a predefined operator from the list shown here in “predefined operator specification” or</li> <li>• a generic operator from the list shown in “generic operator specification”</li> </ul> <p><b>Note:</b> A schema specification is required except for standalone operators.</p>

## DEFINE OPERATOR Statement: Definition

The following example shows a typical DEFINE OPERATOR statement for a standard Teradata PT operator.

The values assigned to the attributes for the DataConnector operator remain in force throughout the job script, except where they are assigned override values in the APPLY statement reference to this operator object.

```
DEFINE OPERATOR FILE_READER
  DESCRIPTION 'TERADATA PARALLEL TRANSPORTER DATACONNECTOR OPERATOR'
  TYPE DATACONNECTOR PRODUCER
  EXTERNAL NAME 'MyOperator'
  SCHEMA ReadFilesSchema
  ATTRIBUTES
  (
    VARCHAR OutputLogFileName,
    VARCHAR FileName,
      Format      = 'Delimited',
    VARCHAR OpenMode     = 'Read',
    VARCHAR FileName,
    VARCHAR IndicatorMode = 'Y',
```

```
    RowsPerInstance = 10000
);
```

# APPLY

## Purpose

An APPLY statement defines the operations to be executed by a job or job step.

Each job step can contain only one APPLY statement.

Each APPLY statement has two parts:

- An APPLY clause or two or more APPLY clauses, separated by commas. Each APPLY clause begins with the keyword APPLY and identifies a data destination and the DML statements to be applied to the source data by the specified consumer operator.  
Followed by:
- A query expression consisting of one or more *query specifications*, each identifying a data source and optionally using a filter operator and optionally using a Teradata PT where clause “filter,” separated by instances of UNION ALL which combine the data sources into a single source data stream.

---

### Note:

If the operator in any APPLY clause is a standalone operator, then any DML statements must be self-contained; that is, they must refer to no data external to the DML statements themselves and there can be no query expression.

---

For examples of typical APPLY statement job script examples, see "IBM z/OS Sample Files" in the *Teradata® Parallel Transporter User Guide*, B035-2445.

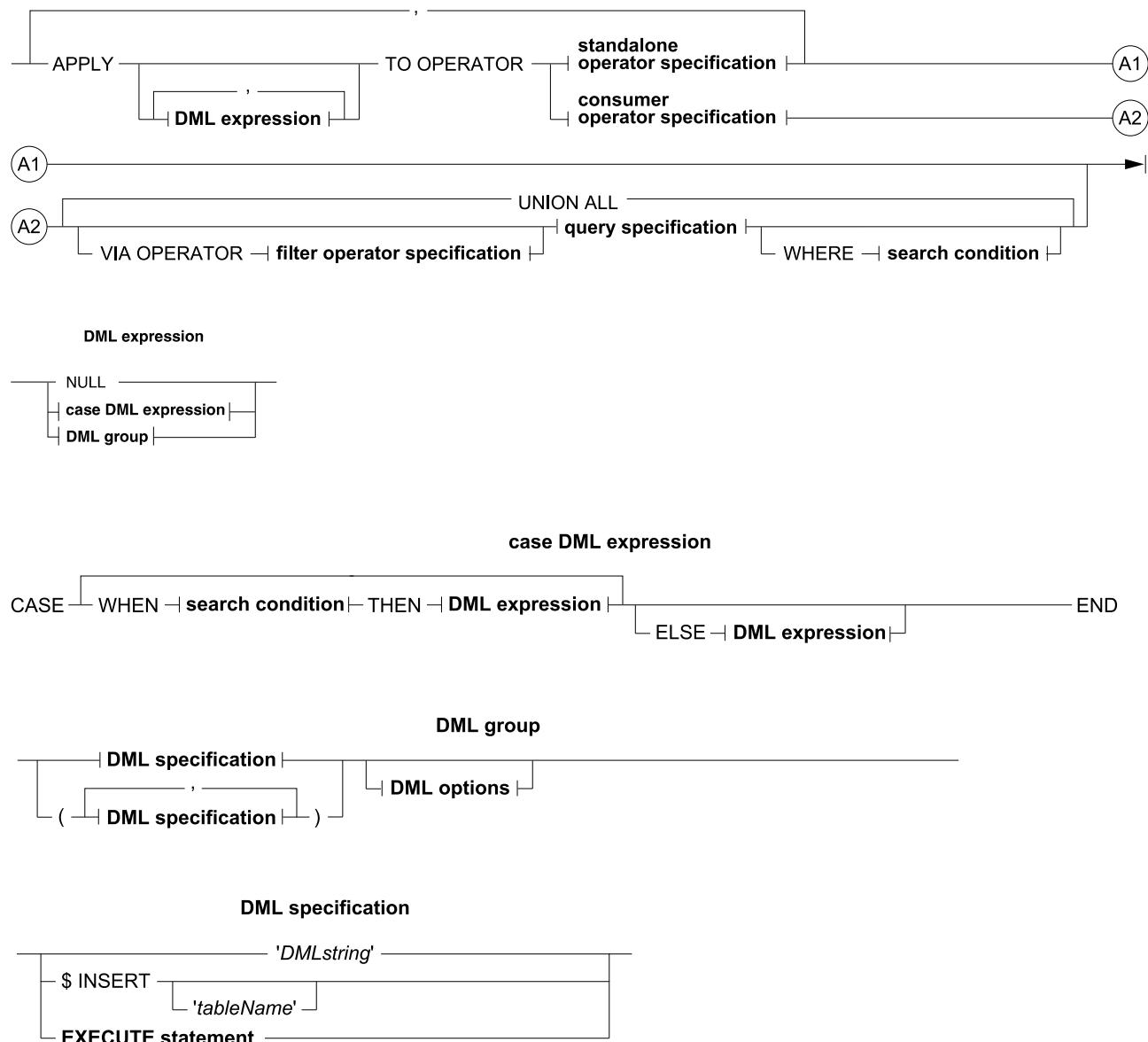
APPLY statements support the following:

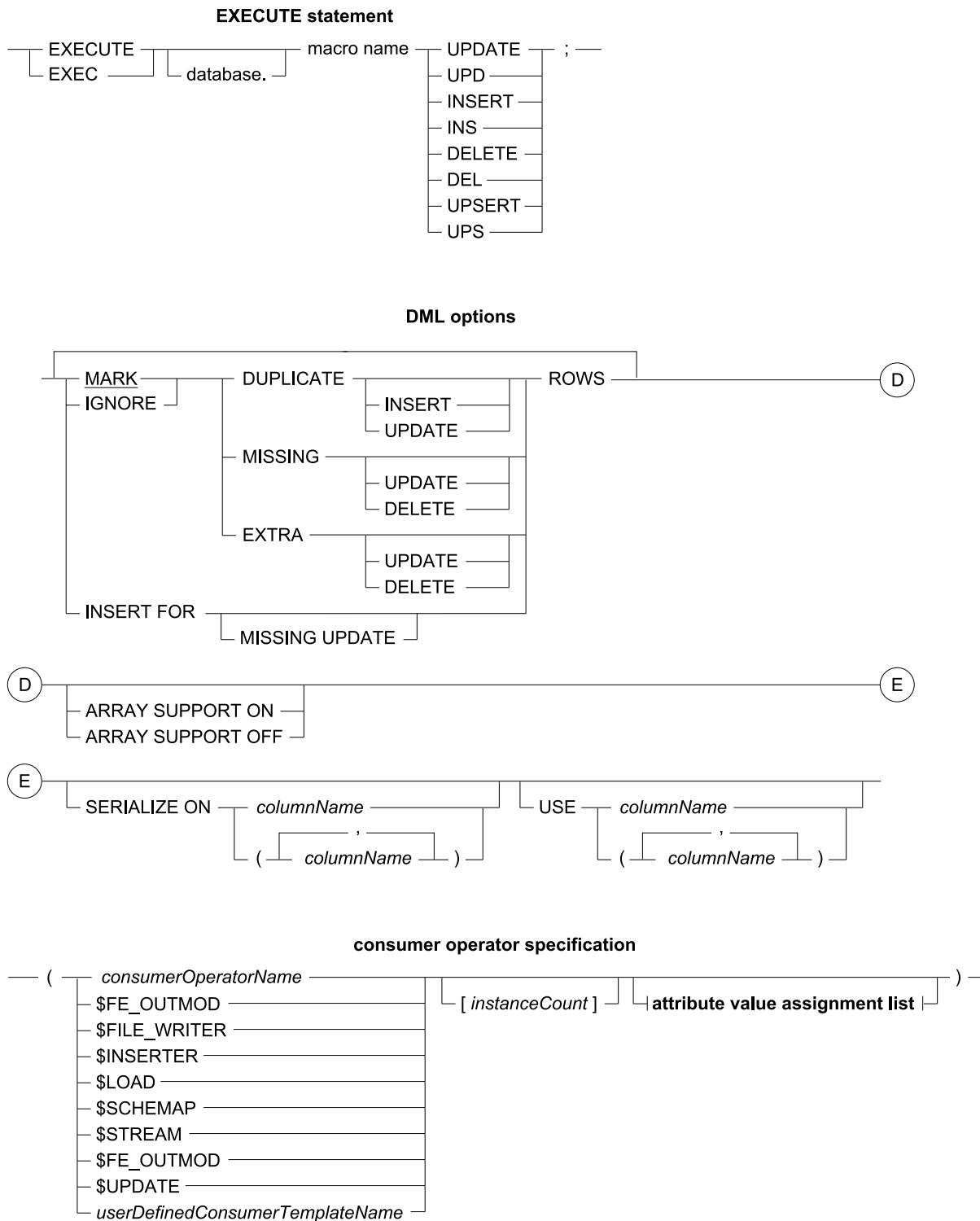
- Multiple data sources using the UNION ALL query.
- Multiple data targets using multiple APPLY clauses.
- Conditional application of DMLs to target tables using CASE DML expressions.
- Optional data filtering using the WHERE clause and VIA filter operators.
- Control of the degree of parallelism by specifying the number of instances of each operator.
- Derivation of new columns using the CASE and SELECT statements.
- A subset of columns from input rows using the SELECT statement.

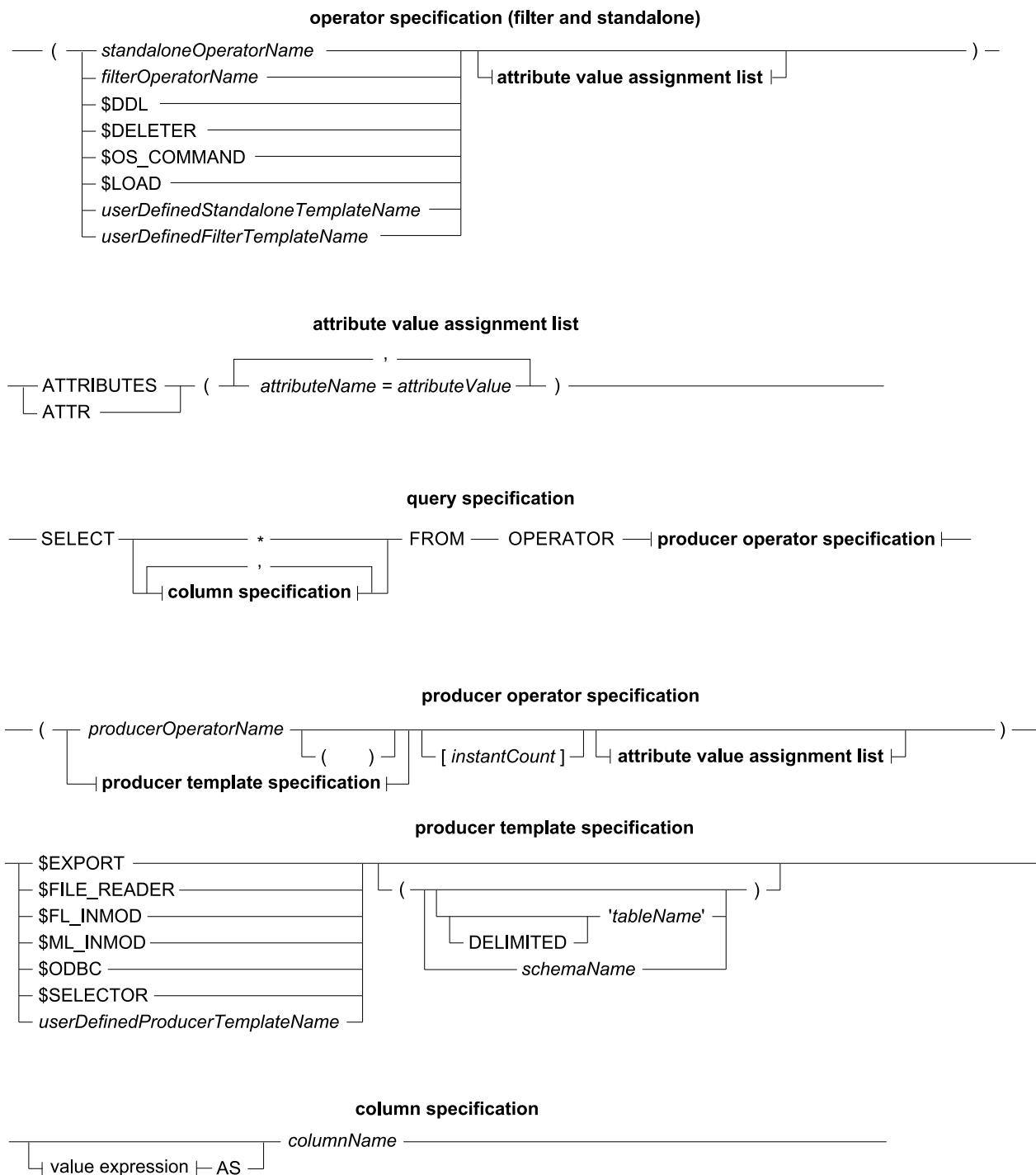
For information about how to implement multiple APPLY operations in a single job step, see *Teradata® Parallel Transporter User Guide*, B035-2445.

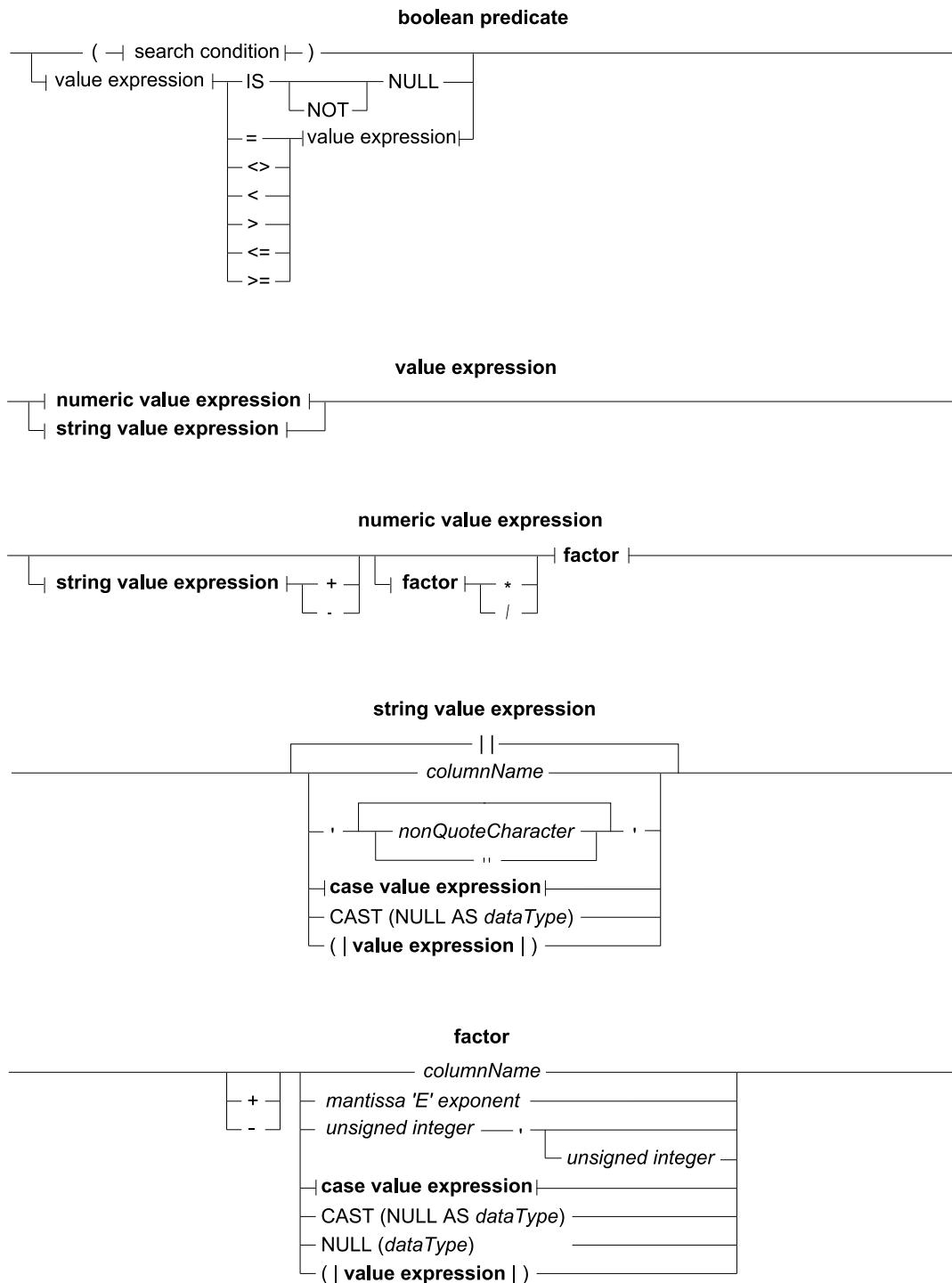
## Syntax

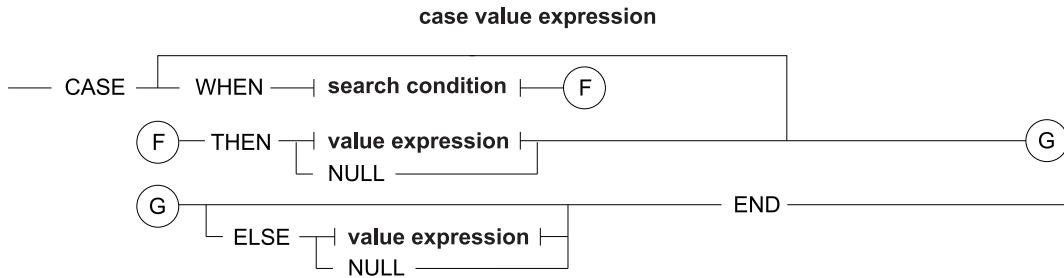
Syntax elements used in an APPLY statement must appear in the order shown in the following syntax diagram.











where:

Syntax Element	Description
*	Indicates that all the columns from the specified data source object must be selected.
	This concatenation operator joins two character strings into one string.
\$DDL	The name of the Teradata PT-provided template for the Data Definition Language (DDL) (standalone) operator.
\$DELETER	The name of the Teradata PT-provided template for the Update Standalone operator used for deleting table rows.
\$EXPORT	The name of the Teradata PT-provided template for the Export (producer) operator.
\$FE_OUTMOD	The name of the Teradata PT-provided template for the FastExport OUTMOD Adapter (consumer) operator.
\$FILE_READER	The more descriptive name of the Teradata PT-provided template for the DataConnector Producer operator.
\$FILE_WRITER	The more descriptive name of the Teradata PT-provided template for the DataConnector Consumer operator.
\$FL_INMOD	The name of the Teradata PT-provided template for the FastLoad INMOD Adapter (producer) operator.
\$INSERT	A Teradata PT job script "placeholder" for an SQL INSERT statement Teradata PT generates based on: <ul style="list-style-type: none"> <li>The column definitions of the identified target table 'tableName', when specified, or</li> <li>A target table that Teradata PT can identify through assignments to various conventional job variables or script assignments to relevant consumer operator attributes.</li> </ul> The generated SQL INSERT statement will replace the \$INSERT placeholder in the job script.
\$INSERTER	The name of the Teradata PT-provided template for the SQL Inserter (consumer) operator.
\$LOAD	The name of the Teradata PT-provided template for the Load (consumer) operator.

Syntax Element	Description
\$ML_INMOD	The name of the Teradata PT-provided template for the MultiLoad INMOD Adapter (producer) operator.
\$ODBC	The name of the Teradata PT-provided template for the ODBC (producer) operator.
\$OS_COMMAND	The name of the Teradata PT-provided template for the OS Command (standalone) operator.
\$SCHEMAP	The name of the Teradata PT-provided template for the Schema Mapping (consumer) operator.
\$SELECTOR	The name of the Teradata PT-provided template for the SQL Selector (producer) operator.
\$STREAM	The name of the Teradata PT-provided template for the Stream (consumer) operator.
\$UPDATE	The name of the Teradata PT-provided template for the Update (consumer) operator.
AND	Designates a Boolean “and” operation.
ARRAY SUPPORT	<p>Indicates whether the Array Support feature will be used for a DML group associated with the Stream operator in the APPLY statement.</p> <ul style="list-style-type: none"> <li>The ON value tells the Stream operator to use the Array Support feature. ON is the default unless the attribute value has been reset.</li> <li>The OFF values tells the Stream operator to not use the Array Support feature. If the ARRAY SUPPORT DML option is specified, it overrides the value of the Stream operator ArraySupport attribute.</li> </ul> <p>If the ARRAY SUPPORT DML option is not specified, the default value for Array Support for the DML group is the value in the Stream operator ArraySupport attribute. Even if no value is specified for the Stream operator ArraySupport attribute, the feature is enabled if basic system requirements for the feature are present.</p> <p>For more information on the Stream operator ArraySupport attribute, see <a href="#">Stream Operator</a>.</p> <p>If the DML statement is an UPSERT statement, it must be specified in the “atomic form.” This method employs a pair of INSERT . . . UPDATE statements, using the INSERT FOR MISSING ROWS option that immediately precedes the Array Support syntax. See <a href="#">Array Support</a>.</p> <p><b>Errors:</b></p> <p>If the ARRAY SUPPORT option is specified without a valid value, the operation will terminate with an error.</p> <p>If the value for ARRAY SUPPORT is set to ‘On’ and either the database or CLIV2 does not support the Array Support feature, the Stream operator will terminate with a fatal error.</p> <p>The Array Support feature only applies to a single DML statement. If the ARRAY SUPPORT DML option is ON, the Stream operator will terminate with an error if it finds multiple DML statements within a single APPLY statement.</p>
AS	Optional. Introduces the column name associated with a derived column, which is a value that results from evaluating an expression.

Syntax Element	Description
<i>attributeName</i>	The name of an operator attribute to which a value is assigned when job execution reaches this point in the job script.
<i>attributeValue</i>	The value that is assigned to the operator attribute just specified, when job execution reaches this point in the job script. <b>Note:</b> <i>attributeValue</i> can also be an array for attributes that support array values. For a description of attributes that support array values, see the topics on individual operators, beginning with <a href="#">DataConnector Operator</a>
ATTRIBUTES ATTR	Optional. Introduces a list of value assignments for the operator referenced in an operator specification.
CASE	In a case expression, CASE introduces one or more conditional expressions where the first (in left-to-right order) value with a true search condition is the value of the CASE expression. If none of the search conditions are true, the CASE expression value is the value of its ELSE expression, if present; otherwise, its value is NULL. In a case DML expression, CASE introduces one or more conditional DML group where the first one (in left-to-right order) with a true search condition is applied to the target of the APPLY statement. If none of the search conditions are true and an ELSE clause is present, then its DML group are applied to the targets; otherwise, no DML groups are applied.
CAST (NULL AS <i>dataType</i> )	This built-in function returns the value NULL in the specified data type.
<i>columnName</i>	Optional column name specification. The column name may already be specified, such as in the SCHEMA attribute of an operator referenced in the APPLY TO or SELECT FROM sections of the APPLY statement.
<i>consumerOperatorName</i>	The name of a consumer operator defined in a DEFINE OPERATOR statement in the job script.
database	Name of the database where the macro is to be executed.
DELETE/DEL	Keyword indicating a DELETE statement is being executed by the macro.
'DMLstring'	Required valid syntax and semantics for SQL INSERT, UPDATE, and DELETE statements.
ELSE	In a case expression, the ELSE keyword introduces an expression where the value is the value of its case expression, if none of the case expression's search conditions are true. In a case DML expression, ELSE introduces a DML group that is applied if none of the search conditions in the case DML expression are true.
ELSE NULL	The optional unconditional default NULL value of a case expression or the optional unconditional default "no action" case of a case DML expression, explicitly specified.

Syntax Element	Description
END	In a case expression, END marks the end of the case expression. In a case DML expression, END marks the end of a CASE group of conditional DML group.
EXECUTE	The EXECUTE statement applies only for the TPT Stream operator.
FALSE	Designates the Boolean “false” value specification.
<i>filterOperatorName</i>	The name of a filter operator defined in a DEFINE OPERATOR statement in the job script.
FROM	Designates the source for the select operation.
INSERT/INS	Keyword indicating an INSERT statement is being executed by the macro.
INSERT FOR	Applies the corresponding INSERT when the row to be UPDATED is missing. Valid only for the Update and Stream operators.  This option applies only when the DML group consists of a single UPDATE statement followed by a single INSERT statement. These keywords are followed by one of the following keywords, which complete the specification of the option (and mean the same thing): <ul style="list-style-type: none"> <li>• ROWS</li> <li>• MISSING UPDATE ROWS</li> </ul> Use the IGNORE MISSING UPDATE ROWS DML option when you use the INSERT FOR ROWS or INSERT FOR MISSING UPDATE option so that missing update rows are not inserted into the error table.
<i>instanceCount</i>	Specifies the number of parallel operator instances. The default is 1. Required to specify multiple instances of the same object in a job step.
IS	Designates a Boolean test.
macro name	Name of the macro resident in the database to be executed.
<i>mantissa 'E' exponent</i>	A numeric constant expressed in standard scientific notation.
MARK or IGNORE	Directs that source rows be put in the error table (MARK) or not put in the error table (IGNORE) when these keywords are associated with certain condition, usually common database errors. <ul style="list-style-type: none"> <li>• When followed by DUPLICATE, MARK directs that a row be put in the error table when it duplicates an existing row in the target table, as follows (valid for Stream and Update operators): <ul style="list-style-type: none"> <li>◦ MARK DUPLICATE ROWS (for both insert and update operations)</li> <li>◦ MARK DUPLICATE INSERT ROWS (for insert operations only)</li> <li>◦ MARK DUPLICATE UPDATE ROWS (for update operations only)</li> </ul> </li> </ul>

Syntax Element	Description
	<p><b>Note:</b></p> <p>A row is a duplicate row if all column values in the row are exactly the same as those of another row. Duplicate row checking is bypassed if the table is a multiset table (which allows duplicate rows), or if the table has one or more unique indexes (the uniqueness tests) make any duplicate row check unnecessary) ; in these cases, IGNORE DUPLICATE ROWS has no effect. Any uniqueness violations result in the offending rows going to the error table.</p> <ul style="list-style-type: none"> <li>When followed by MISSING, MARK directs that a row be put in the error table when the row it targets for modification or deletion does not exist in the target table, as follows (valid for Stream and Update operators): <ul style="list-style-type: none"> <li>MARK MISSING ROWS (for both update and delete operations)</li> <li>MARK MISSING UPDATE ROWS (for update operations only)</li> <li>MARK MISSING DELETE ROWS (for delete operations only)</li> </ul> </li> <li>When followed by EXTRA, MARK directs that a row be put in the error table when the row modifies or deletes multiple rows in the target table, as follows (valid for Stream operator): <ul style="list-style-type: none"> <li>MARK EXTRA ROWS (for both update and delete operations)</li> <li>MARK EXTRA UPDATE ROWS (for update operations only)</li> <li>MARK EXTRA DELETE ROWS (for delete operations only)</li> </ul> </li> </ul> <p>For all these cases, if IGNORE is specified instead of MARK, it means that the row should not be put in the error table.</p>
<i>nonQuote Character</i>	Any character other than the single-quote ('') character
NOT	Designates the Boolean negation.
NULL	Designates that no element is to be applied.
NULL ( <i>dataType</i> )	Same as CAST (NULL as <i>dataType</i> ).
OPERATOR	Designates a Teradata PT operator.
OR	Designates a Boolean “or” operation.
<i>producerOperatorName</i>	The name of a producer operator defined in a DEFINE OPERATOR statement in the job script.
<i>schemaName</i>	The name of a Teradata PT schema defined in a DEFINE SCHEMA statement in the job script.
SELECT	Designates the select option of a Teradata PT data movement operation.
SERIALIZE ON	Invokes the serialization feature, which guarantees that operations on a specified column occur serially. Valid only for the Stream operator.
<i>standaloneOperatorName</i>	The name of a standalone operator defined in a DEFINE OPERATOR statement in the job script.

Syntax Element	Description
<i>tableName</i>	The name of an existing table whose column definitions will be the basis for a Teradata PT-generated DEFINE SCHEMA statement.
THEN	In a case expression, THEN introduces the value expression corresponding to a search condition of a conditional expression. In a case DML expression, THEN introduces a DML group that is applied when the search condition of the corresponding WHEN keyword is the first true condition encountered.
TO	Designates the targets of an update operation.
TRUE	Designates the Boolean “true” value specification.
UNION ALL	Enables combination and parallel processing of data from multiple sources.
UNKNOWN	Designates the value of a Boolean predicate whose truth or falsity cannot be determined because one or more operands are NULL-valued.
<i>unsigned integer</i>	An integer that has no preceding “+” or “-” sign.
UPDATE/UPD	Keyword indicating an UPDATE statement is being executed by the macro.
UPSERT/UPS	Keyword indicating an Atomic upsert is being executed by the macro.
USE	Introduces the names of the columns transmitted to the database when the SQL statements in a DML group are executed during APPLY processing. Valid only for the Stream operator.
<i>userDefinedConsumerTemplateName</i>	The name of a user-created consumer operator template.
<i>userDefinedFilterTemplateName</i>	The name of a user-created filter operator template.
<i>userDefinedProducerTemplateName</i>	The name of a user-created producer operator template.
<i>userDefinedStandaloneTemplate</i>	The name of a user-created standalone operator template.
VIA	<p>Specifies the filter operator that filters/modifies/alters data after it is processed by a producer operator.</p> <p>The VIA clause goes between the APPLY keyword and the SELECT keyword:</p> <pre> APPLY ( ) TO OPERATOR ( consumer_operator [ n ] ) VIA OPERATOR (filter_operator [n]) SELECT... FROM OPERATOR (producer_operator [n]); </pre>

# DataConnector Operator

## DataConnector Operator Capabilities

The DataConnector operator can either:

- Read data from flat files, access modules, or Hadoop files and tables. As a reader, it is considered a producer operator, that is, one that produces a data stream.
- Write data to flat files, access modules, or Hadoop files and tables. As a writer, it is considered a consumer operator, that is, one that consumes a data stream.

The DataConnector operator can also be used to scan directories for files to be processed; in the TPT documentation, this is referred to as *Batch Directory Scan*. Scanning can be done in a continuous manner based on time intervals; in the TPT documentation, this is referred to as *Active Directory Scan*. For more information, see the *Teradata® Parallel Transporter User Guide*, B035-2445.

The Active Directory Scan functionality is supported when using the HDFS interface to process Hadoop files.

The Active Directory Scan functionality is not supported when using the TDCH-TPT interface to process Hadoop files and tables.

For more information about the DataConnector Operator's Hadoop interfaces, see [Processing Hadoop Files and Tables](#).

Parallel processing can be accomplished by specifying multiple instances of the operator.

## Syntax for the DataConnector Operator

Use the following required and optional specifications to define the DataConnector operator in a Teradata PT job script.

## Required Specifications

The following specifications are required to define a DataConnector job. Note that certain platforms require additional operator specifications.

### Required Syntax for the DataConnector Operator

Specification	Description
TYPE	Operator type. Either DATACONNECTOR PRODUCER or DATACONNECTOR CONSUMER, depending on the direction of data flow to/from the external device: <ul style="list-style-type: none"> <li>• DATACONNECTOR PRODUCER = reading data from an external device/file.</li> <li>• DATACONNECTOR CONSUMER = writing data to an external device/file.</li> </ul>
FileName	Operator attribute specifying the name(s) of the file(s).

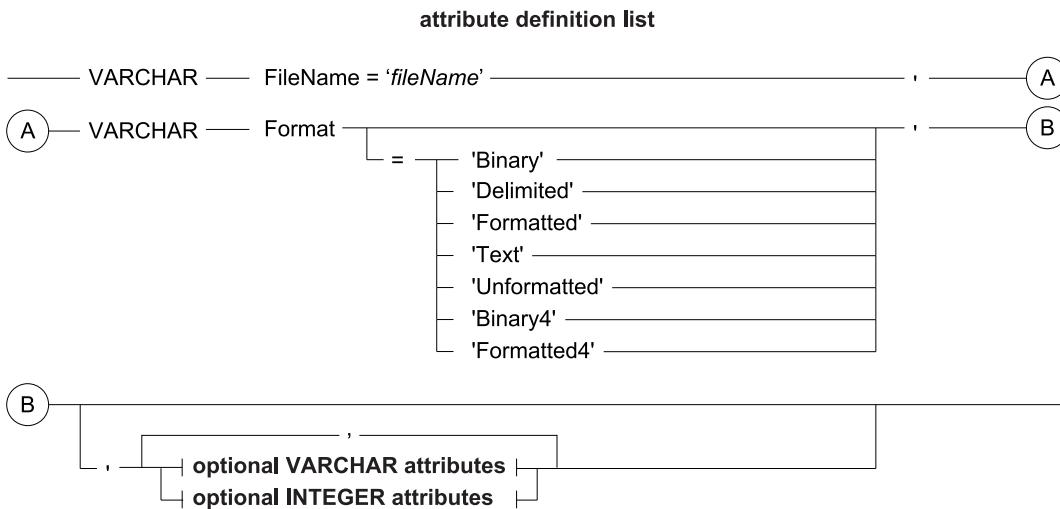
Specification	Description
	On z/OS, a DDNAME can be indicated by including a "DD" prefix.
Format	Operator attribute specifying the logical record format of the file being read.

## Required and Optional Attributes

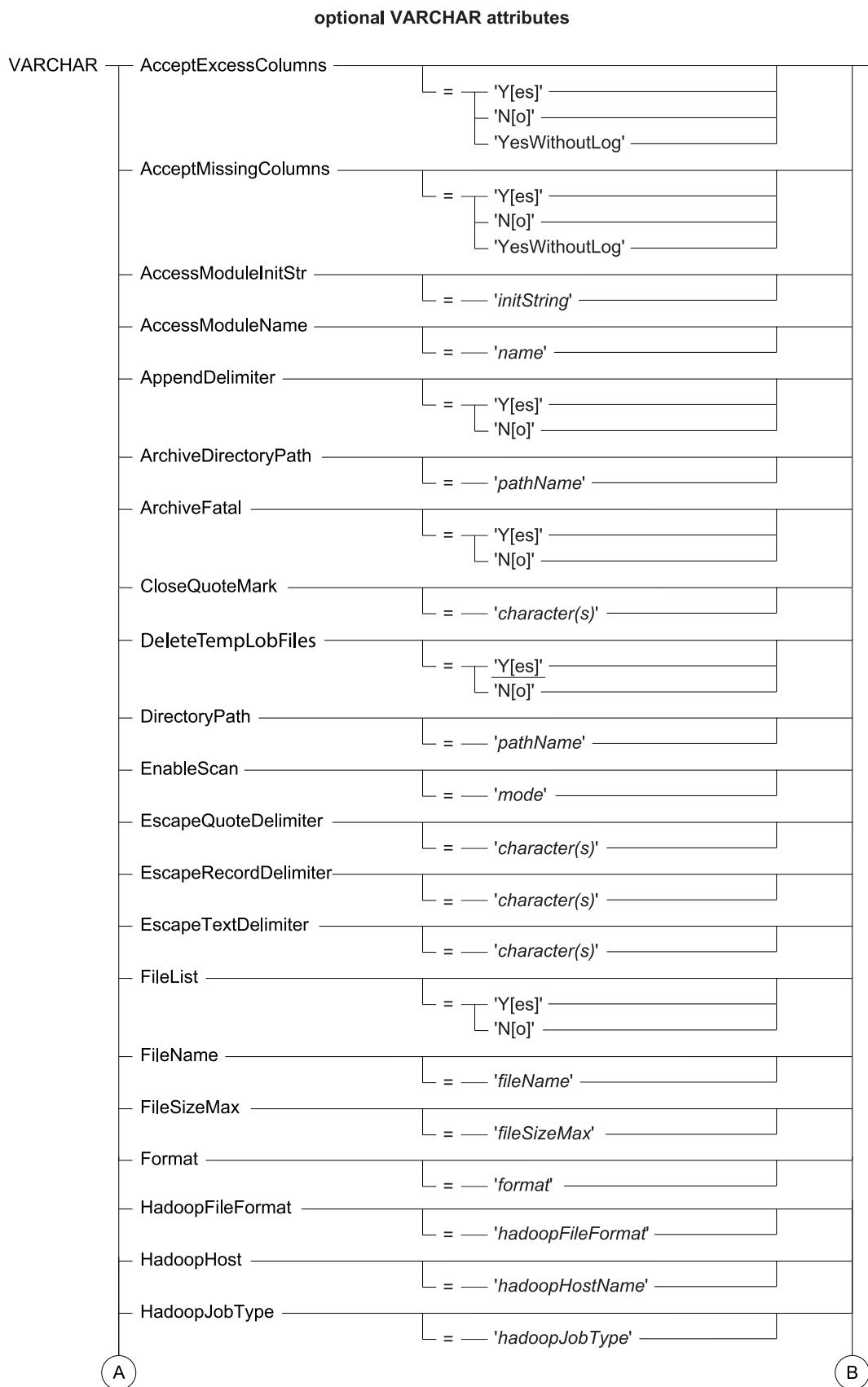
Use the attribute definition list syntax in the Teradata PT DEFINE OPERATOR statement to declare the required and optional attribute values for the DataConnector operator.

Parallel processing of multiple files is permitted. Multiple instances of the producer DataConnector operator are allowed by specifying a base directory in the DirectoryPath attribute, and then a wildcard in the FileName attribute as a selection basis for a series of files to be read.

The specification of any attributes that begin with 'Hadoop' will cause the DataConnector operator to process Hadoop files, directories, and tables, rather than files and directories in the local filesystem. For more information, see [Processing Hadoop Files and Tables](#).



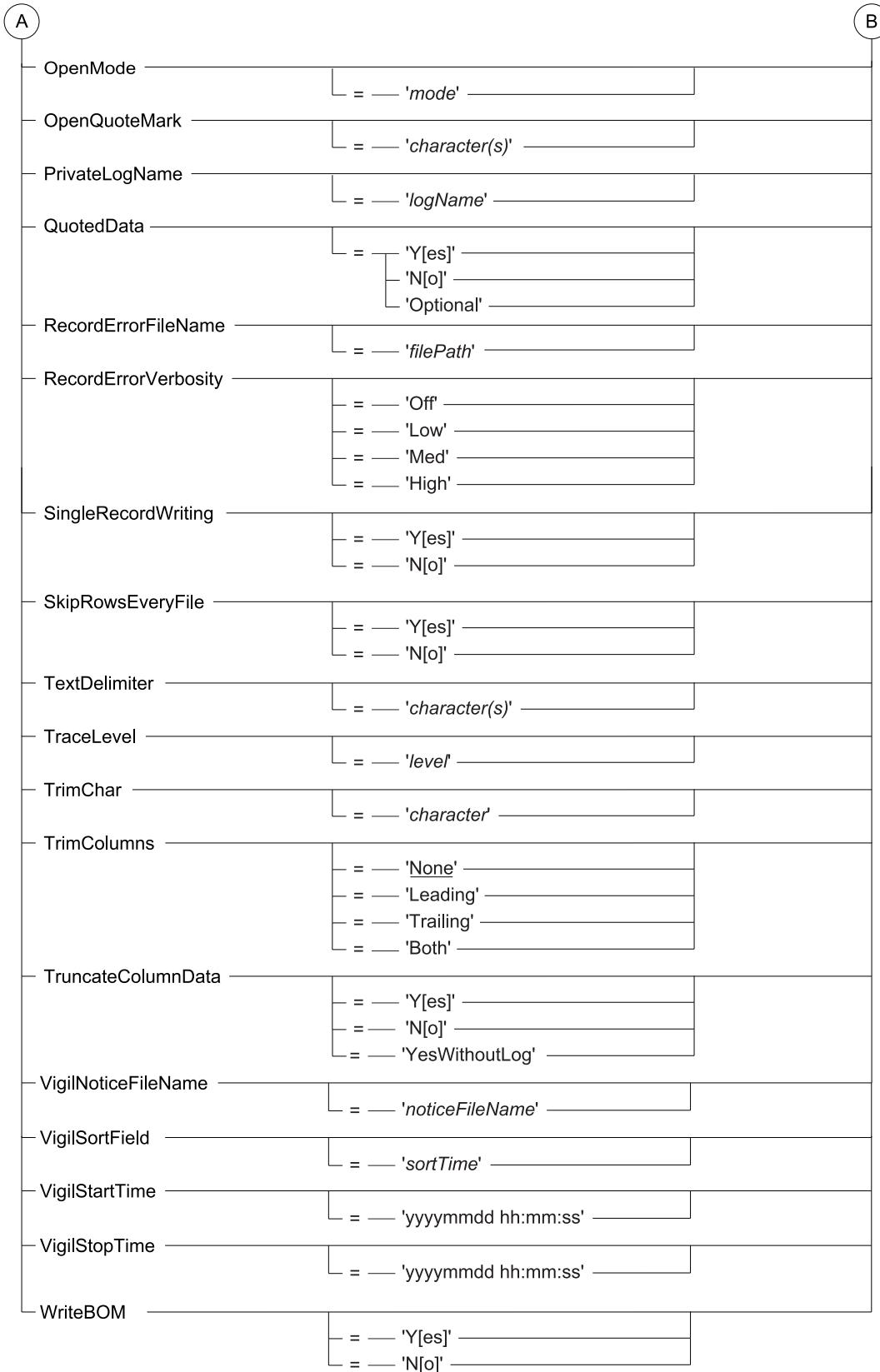
optional INTEGER attributes	
INTEGER	ErrorLimit = <i>errorLimit</i>
	HadoopBlockSize = ( <i>X</i> * 1K Bytes)
	HadoopNumMappers = <i>hadoopNumMappers</i>
	NamedPipeTimeOut = <i>seconds</i>
	RecordsPerBuffer = <i>RecordCount</i> (optional)
	RowsPerinstance = <i>rows</i>
	SkipRows = <i>numberOfRows</i>
	Timeout = <i>seconds</i>
	VigilElapsedTime = <i>elapsedMinutes</i>
	VigilMaxFiles = <i>numberOfFiles</i>
	VigilWaitTime = <i>waitSeconds</i>



## optional VARCHAR attributes (continued)



## optional VARCHAR attributes (continued)



where:

#### DataConnector Attribute Descriptions

Syntax Element	Description
AcceptExcessColumns = ' <i>option</i> '	<p>Optional attribute that specifies whether or not rows with extra columns are acceptable. This only applies to delimited data for the producer operator.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>• 'Y[es]' = A row with extra columns is truncated to the number of columns defined in the schema, and sent downstream without an error being raised. The edited record is sent to the database and the original record is saved in the record error file, if it is defined via the RecordErrorFileName attribute.</li> <li>• 'N[o]' = A row with extra columns is not sent to the database, but the original record is saved to the record error file, if it is defined using the RecordErrorFileName attribute. If RecordErrorFileName is not defined, a fatal error will occur. 'No' is the default value.</li> <li>• 'YesW[ithoutLog]' = The edited row is sent to the database, but the original record is not saved in the record error file.</li> </ul> <p>This attribute is ignored by the consumer operator.</p>
AcceptMissingColumns = ' <i>option</i> '	<p>Optional attribute that determines how rows are treated in which the column count is less than that defined in the schema. This only applies to delimited data for the producer operator.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>• 'Y[es]' = The row is to be extended to the correct number of columns. Each appended column will be a zero length column and be processed according to the value of the NullColumns attribute. The edited record is sent to the database and the original record is saved in the record error file, if it is defined via the RecordErrorFileName attribute.</li> <li>• 'N[o]' = A row with too few columns is not sent to the database, but the original record is saved in the record error file, if it is defined via the RecordErrorFileName attribute. If RecordErrorFileName is not defined, a fatal error will occur. 'No' is the default value.</li> <li>• 'YesW[ithoutLog]' = The edited row is sent to the database, but the original record is not saved in the record error file.</li> </ul> <p>This attribute is ignored by the consumer operator.</p>
AccessModuleInitStr = ' <i>initString</i> '	Optional attribute that specifies the initialization string for the specified access module. This

Syntax Element	Description
	<p>attribute is used by both the producer and consumer operators.</p> <p>If the AccessModuleInitStr attribute is defined and the Filename attribute is not, then the Open and Read requests will be sent to the access module of each instance. However, the filename passed to the access module will be an empty string, so it is up to the access module itself to determine (possibly from the initialization string) which file is to be opened. If any Open or Read requests fail within the access module, the job will be terminated.</p> <p>For the <i>initString</i> values, see the Initialization String section for each module in <i>Teradata® Tools and Utilities Access Module Reference</i>, B035-2425.</p>
AccessModuleName = 'name'	<p>Optional attribute that specifies the name of the access module file, where the value for <i>name</i> is dependent on the following:</p> <p><b>Teradata Access Module for S3</b></p> <ul style="list-style-type: none"> <li>• libstsaxsmod.so on Linux platform</li> </ul> <p><b>Teradata Access Module for Named Pipes</b></p> <ul style="list-style-type: none"> <li>• libnp_axsmod.dylib on the Apple macOS platform</li> <li>• np_axsmod.so on all other UNIX platforms</li> <li>• np_axsmod.dll on Windows platforms</li> </ul> <p><b>Teradata Access Module for WebSphere MQ (client version)</b></p> <ul style="list-style-type: none"> <li>• libmqsc.dylib on the Apple macOS platform</li> <li>• libmqsc.so on all other UNIX platforms</li> <li>• libmqsc.dll on Windows platforms</li> </ul> <p><b>Teradata Access Module for WebSphere MQ (server version)</b></p> <ul style="list-style-type: none"> <li>• libmqsv.dylib on the Apple macOS platform</li> <li>• libmqsv.so on all other UNIX platforms</li> <li>• libmqsv.dll on Windows platforms</li> </ul> <p><b>Teradata Access Module for OLEDB</b></p> <ul style="list-style-type: none"> <li>• oledb_axsmod.dll on Windows platforms</li> </ul> <p><b>Teradata Access Module for Kafka</b></p> <ul style="list-style-type: none"> <li>• libkafkaaxsmod.so on Linux platform</li> </ul> <p><b>Teradata Access Module for Azure Blob</b></p> <ul style="list-style-type: none"> <li>• libazureaxsmod.so on Linux platform</li> </ul> <p><b>Teradata Access Module for Google Cloud Storage (GCS)</b></p> <ul style="list-style-type: none"> <li>• libgcsaxsmod.so on Linux platform</li> </ul> <p>Use your shared library file name if you use a custom access module.</p>

Syntax Element	Description
	<p>Access module names do not need a suffix since the operator appends the correct suffix for the platform used.</p> <p>If a path is not included in the access module name, Teradata PT will search for it in the following order:</p> <ul style="list-style-type: none"> <li>• Search the sub-directory within the installation directory that stores the TPT libraries.</li> <li>• Search the current working directory.</li> <li>• Allow the operating system to search default system directories or paths defined through linker environment variables (such as LD_LIBRARY_PATH).</li> </ul> <p>This attribute is used by both the producer and consumer operators.</p> <p><b>Note:</b></p> <p>Large File Access Module is no longer available because the DataConnector operator now supports file sizes greater than 2 gigabytes on Windows, AIX, and Solaris running on SPARC systems when system parameters are appropriately set.</p>
AppendDelimiter = ' <i>option</i> '	<p>Optional attribute that adds a delimiter at the end of every record written. Use <i>AppendDelimiter</i> when creating delimited output files with the consumer operator.</p> <p>When the last column in the record is NULL, a trailing delimiter denotes that the column is NULL.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>• 'Y[es]' = Adds a delimiter at the end of every record written.</li> <li>• 'N[o]' = Does not add a delimiter at the end of every record written (default).</li> </ul> <p>This attribute is not valid for the producer operator.</p>
ArchiveDirectoryPath = ' <i>pathName</i> '	<p>Defines the complete pathname of a directory to which all processed files are moved from the current directory (specified with the DirectoryPath attribute) for the producer operator.</p> <p>This attribute is required when specifying a value for the VigilMaxFiles attribute.</p> <p>This attribute is ignored by the consumer operator.</p>
ArchiveFatal = ' <i>option</i> '	<p>Defines what action to take if an archive (file move) fails for the producer operator.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>• 'Y[es]' = The job terminates (default).</li> <li>• 'N[o]' = Processing continues with a warning.</li> </ul> <p>This attribute is ignored by the consumer operator.</p>

Syntax Element	Description
CloseQuoteMark = ' <i>character(s)</i> '	<p>Defines the character sequence for the closing quote mark within delimited data.</p> <p>May be any single or multibyte value from the session character set. For example, '' '' or '    '.</p> <p>The default value is the sequence provided for the attribute OpenQuoteMark.</p> <p>This attribute is used by both the producer and consumer operators.</p>
DeleteTempLobFiles	<p>Optional attribute that tells the operator whether or not to delete the temporary LOB directory and the temporary LOB files in the directory at cleanup. This only applies for the consumer operator (file writer).</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>'Y[es]' = Tells the operator to delete the temporary LOB directory at cleanup. This is the default.</li> <li>'N[o]' = Tells the operator not to delete the temporary LOB directory at cleanup. The user is responsible for deleting the temporary LOB directory.</li> </ul> <p><b>Note:</b></p> <p>When the temporary LOB files exist in a remote NFS mount directory, the suggestion is to set this attribute to 'No' and the operator will not spend time deleting the temporary LOB files.</p> <p><b>Note:</b></p> <p>Temporary LOB files will be created in the temporary LOB directory.</p> <p>The naming convention for the temporary LOB directory will be as follows:</p> <ul style="list-style-type: none"> <li>When the producer's LobDirectoryPath attribute is set, the name of the temporary LOB directory will be this: &lt;LobDirectoryPath&gt;/&lt;job id&gt;_s&lt;job step number&gt;_TempLOBDir</li> <li>When the producer's LobDirectoryPath attribute is not set, the name of the temporary LOB directory will be this: &lt;current working dir&gt;/&lt;job id&gt;_s&lt;job step number&gt;_TempLOBDir</li> </ul> <p>This attribute is not supported on z/OS.</p>
DirectoryPath = ' <i>pathName</i> '	<p>This optional attribute defines the location of source files for reading (by the producer operator) or target files for writing (by the consumer operator).</p> <p>Use this attribute to specify an existing base directory path (or z/OS PDS dataset name) for the location of the file (or PDS members) indicated by the FileName attribute. This attribute cannot be</p>

Syntax Element	Description
	<p>used if a z/OS data set (DD:DATA) is specified in the FileName attribute.</p> <p>To specify an z/OS PDS data set with a JCL DD statement, prefix the DirectoryPath attribute value with 'DD:' as shown in the following example:</p> <pre>DirectoryPath='DD:&lt;ddname&gt;'</pre> <p>To specify the z/OS PDS data set directly, use the following syntax:</p> <pre>DirectoryPath='//dataset-name'</pre> <p>This attribute defaults to the directory in which the job is executing (the job working directory specified in the DEFINE JOB statement).</p> <p>If the DataConnector is a producer instance, the Directory Path specification is prepended to the file name only if no directory names appear within the FileName attribute. If a directory is included in the FileName attribute, then the DirectoryPath attribute is expected to be empty.</p>
EnableScan = ' <i>mode</i> '	<p>Optional attribute that bypasses the directory scan logic when using access modules with the producer operator.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>• 'Y[es]' = Operator retains its original behavior, which is to automatically scan directories (default).</li> <li>• 'N[o]' = Operator bypasses the directory scan feature and passes directly to the access module only the file specified in the FileName attribute.</li> </ul> <p>If this attribute is set to 'No' while a wildcard character is specified in the FileName attribute, a warning message is generated in the DataConnector log.</p> <p>This attribute is not valid for the consumer operator.</p>
ErrorLimit = <i>errorLimit</i>	<p>errorLimit = (0 - 2147483647) 0 = Default (Unlimited)</p> <p>Optional attribute that specifies the approximate number of records that can be stored in the error row file before the DataConnector producer operator job is terminated. If ErrorLimit is not specified, it is the same as an ErrorLimit value of 0. The ErrorLimit specification applies to each instance of the DataConnector producer operator.</p> <p>When the "RecordErrorFileName" attribute is defined (previously known as "RowErrFileName"), error records are saved in the specified file and the</p>

Syntax Element	Description
	<p>job continues to process additional records without exiting with a fatal error.</p> <p>This attribute is ignored by the consumer operator. For information about the effects of the ErrorLimit attribute, see <i>Teradata® Parallel Transporter User Guide</i>, B035-2445.</p> <p><b>Note:</b></p> <p>For a list of obsolete syntax, which are supported but no longer documented, see <a href="#">Deprecated Syntax</a>.</p>
EscapeQuoteDelimiter = ' <i>character(s)</i> '	<p>Optional attribute that allows you to define the escape quote character sequence within delimited data. The default value is the sequence provided for the CloseQuoteMark attribute. See <a href="#">Rules for Quoted Delimited Data Handling</a>.</p> <p>When processing data in delimited format, if the EscapeQuoteDelimiter precedes either the OpenQuoteMark or the CloseQuoteMark, that instance of the quote mark (either open or close) is included in the data rather than marking the beginning or end of a quoted string.</p> <p>This attribute is used by both the producer and consumer operators.</p>
EscapeRecordDelimiter = ' <i>character(s)</i> '	<p>Optional attribute that allows you to define the record delimiter escape sequence within unquoted delimited data.</p> <p>When processing data in delimited format, if the escape sequence defined by EscapeRecordDelimiter precedes the end-of-record (EOR) marker, the escape sequence is removed and that instance of the EOR is included in the data rather than marking the end of the record. If the escape sequence defined by EscapeRecordDelimiter is not immediately followed by an EOR, the escape sequence is treated as regular data.</p> <p>Other details:</p> <ul style="list-style-type: none"> <li>• ArchiveFatal, DataConnectorAn EOR is either a LF (0x0A), CRLF (0x0D0A), or an EBCDIC newline (0x15).</li> <li>• EscapeRecordDelimiter is ignored for all format types other than 'Delimited'.</li> <li>• EscapeRecordDelimiter can be one or more characters in length and can contain single-byte or multi-byte characters depending upon the session character set.</li> </ul>

Syntax Element	Description
	<ul style="list-style-type: none"> <li>For quoted data, an EscapeRecordDelimiter is treated as regular data, just like an embedded EOR.</li> <li>EscapeRecordDelimiter is only used by the producer operator. It is ignored by the consumer operator.</li> <li>There is no default value.</li> </ul>
EscapeTextDelimiter = ' <i>character(s)</i> '	<p>Optional attribute that allows you to define the delimiter escape character sequence within delimited data.</p> <p>When processing data in delimited format, if the escape sequence defined by EscapeTextDelimiter precedes the delimiter, that instance of the delimiter is included in the data rather than marking the end of the column. If the escape sequence defined by EscapeTextDelimiter is not immediately followed by the delimiter character, the data is considered to be ordinary and no further processing is performed.</p> <p>For example, if the default delimiter is the pipe (   ) and the EscapeTextDelimiter is the backslash, then column data input of abc def] would be loaded as abc def.</p> <p>Other details:</p> <ul style="list-style-type: none"> <li>EscapeTextDelimiter is ignored for all format types other than 'Delimited'.</li> <li>EscapeTextDelimiter can be one or more characters in length and can contain single-byte or multi-byte characters depending upon the session character set.</li> <li>For quoted data, an EscapeTextDelimiter is treated as regular data.</li> <li>EscapeTextDelimiter is used by both the producer and consumer operators.</li> <li>There is no default value.</li> </ul>
FileList = ' <i>option</i> '	<p>Optional attribute used in conjunction with the FileName attribute.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>'Y[es]' = The file specified by FileName contains a list of files to be processed. For the producer operator, any number of files can be listed, each being read for data. For the consumer operator, the number of files listed must equal the number of consumer instances, so that each instance has a file to write to.</li> <li>'N[o]' = The file specified by FileName does not contain a list of files to be processed.</li> </ul>

Syntax Element	Description
	<p><b>Note:</b></p> <p>The DataConnector operators only support a fileList file encoded in ASCII on network-attached platforms.</p>
FileName = ' <i>fileName</i> '	<p>Required attribute that specifies the name of the file to be processed. For the producer operator, this file defines where to read source data. For the consumer operator, it defines where to write target data.</p> <p>In some cases, the access module specified using the AccessModuleName attribute may not use or recognize file names and, therefore, may not require specification of a value for the FileName attribute. For example, the Teradata Access Module for IBM Websphere MQ does not require a file name specification.</p> <p>When used with the fileList attribute, <i>fileName</i> is expected to contain a list of names of the files to be processed, each with a full path specification. In this case, wildcard characters are not supported for either the FileName attribute or the filenames it contains. Multiple instances of the operator can be used to process the list of files in parallel.</p> <p>When fileList='No', wildcard characters in the <i>fileName</i> will result in a directory scan by the producer operator. However, on Windows platforms, using the wildcard character (*) in the '<i>filename</i>' operator attribute may inadvertently include more files than you desire. For example, if you specify *.dat, a directory scan of the folder will find files as if you had specified*.dat*; for example, files with the extension .data, .date, and .dat071503 will also be found. Therefore, you may need to first remove extraneous files from your folder.</p> <p>Reading and writing of a GZIP compressed file is supported on all z/OS platforms. The support for this is enabled automatically based on the file extension. The standard file name extension for gzip files is ".gz".</p> <p>Reading and writing of a ZIP compressed file is supported on Windows and Unix, but not on IBM z/OS. The support for this is enabled automatically based on the file extension. The standard file name extension for zip files is ".zip".</p> <p>Only single files are supported with the ZIP format for both reading and writing.</p> <p>Reading and writing of GZIP and ZIP files is not supported when using Hadoop/HDFS.</p>

Syntax Element	Description
	<p>For additional z/OS dataset syntax, see the table <a href="#">Valid Filename Syntax</a>.</p>
FileSizeMax = ' <i>fileSizeMax</i> '	<p>Optional attribute which is only valid for a consumer instance. It defines the maximum file size (in bytes) for an output file. When a file reaches this size, DataConnector closes the file and continues writing records to a new output file, using the original user-defined file name with an appended instance number and file number.</p> <p>Valid values for this attribute are:</p> <ul style="list-style-type: none"> <li>Any positive integer within the file size limit enforced by the current file system.</li> <li>A shorthand designation consisting of leading digits followed by a 'K', 'M', or 'G', which represent 1024 bytes, 1048576 bytes, and 1073741824 bytes, respectively. For example, '350M' represents 367001600 bytes. The overall numeric representation must be within the file size limit enforced by the current file system.</li> </ul> <p>The naming of output files is based on the FileName setting, the instance number, and the number of files that have already been opened for that instance. The specific naming syntax is the base file name taken from the FileName setting, followed by a hyphen ('-'), followed by the operator instance number, followed by another hyphen, followed by a numerical count (file number), and then followed by the file extension (if it exists) from the FileName setting. For example, if 3 instances of DataConnector are running and FileName='abcd.txt', the following 9 files may be created:</p> <ul style="list-style-type: none"> <li>abcd-1-1.txt – first file for first instance</li> <li>abcd-2-1.txt – first file for second instance</li> <li>abcd-3-1.txt – first file for third instance</li> <li>abcd-1-2.txt – second file for first instance</li> <li>abcd-2-2.txt – second file for second instance</li> <li>abcd-3-2.txt – second file for third instance</li> <li>abcd-1-3.txt – third file for first instance</li> <li>abcd-2-3.txt – third file for second instance</li> <li>abcd-3-3.txt – third file for third instance</li> </ul> <p>This naming convention is always used when the FileSizeMax attribute is defined, no matter if just one operator instance is used or if the size limit is never reached.</p> <p>Individual records will not span files. So, the final size of a maxed-out file may be slightly less than the FileSizeMax value.</p> <p>The DataConnector consumer operator must be able to write at least one record to each file. So,</p>

Syntax Element	Description
	<p>it is possible for a file to exceed the FileSizeMax value, only if the file contains a single record and that record exceeds the FileSizeMax limit.</p> <p>When using multiple instances, some instances may produce more output files than other instances, simply because they are processing more records from the data stream. To ensure a more evenly distribution of records across files for each instance, use tbuild's -C command line option.</p> <p>The FileSizeMax attribute is not supported for the following:</p> <ul style="list-style-type: none"> <li>• The DataConnector producer operator</li> <li>• The z/OS platform</li> <li>• ZIP and GZIP files</li> <li>• Hadoop files</li> <li>• FastExport OUTMOD, FastLoad INMOD, and MultiLoad INMOD operators</li> </ul> <p>The default value (0) indicates that no file size limit will be enforced.</p>
Format = ' <i>format</i> '	<p>Required attribute that specifies the logical record format of the data. This attribute is used by both the producer operator and the consumer operator. No system default exists.</p> <p>Format can have any of the following values:</p> <ul style="list-style-type: none"> <li>• 'Binary' = 2-byte integer, n, followed by n bytes of data. This data format requires rows to be 64KB (64260 data bytes) or smaller. In this format:           <ul style="list-style-type: none"> <li>◦ The data is prefixed by a record-length marker.</li> <li>◦ The record-length marker does not include the length of the marker itself.</li> <li>◦ The record-length is not part of the transmitted data.</li> </ul> </li> <li>• 'Binary4' = 4-byte integer, n, followed by <i>n</i> bytes of data. This data format supports rows up to 1MB (1000000 data bytes) in size. In this format:           <ul style="list-style-type: none"> <li>◦ The data is prefixed by a record-length marker.</li> <li>◦ The record-length marker does not include the length of the marker itself.</li> <li>◦ The record-length is not part of the transmitted data.</li> </ul> </li> <li>• 'Delimited' = in text format with each field separated by a delimiter character. When you specify Delimited format, you can use the optional TextDelimiter attribute to specify the delimiter character. The default is the pipe character (   ).</li> </ul>

Syntax Element	Description
	<p><b>Note:</b></p> <p>When the Format attribute of the DataConnector Producer is set to 'delimited', the associated Teradata PT schema object must be comprised of only VARCHAR and/or VARDATE columns. Specifying non-VARCHAR or non-VARDATE columns results in an error.</p> <ul style="list-style-type: none"> <li>'Formatted' = both prefixed by a 2-byte record-length marker and followed by an end-of-record marker. This data format requires rows to be 64KB (64260 data bytes) or smaller. In this format: The record-length marker does not include the length of the marker itself. Neither the record-length nor the end-of-record marker is part of the transmitted data.</li> <li>'Formatted4' = both prefixed by a 4-byte record-length marker and followed by an end-of-record marker. This data format supports rows up to 1MB (1000000 data bytes) in size. In this format: The record-length marker does not include the length of the marker itself. Neither the record-length nor the end-of-record marker is part of the transmitted data.</li> <li>'Text' = character data separated by an end-of-record (EOR) marker. The EOR marker can be either a single-byte linefeed (X'0A') or a double-byte carriage-return/line-feed pair (X'0D0A'), as defined by the first EOR marker encountered for the first record. This format restricts column data types to CHAR or ANSIDATE only.</li> <li>'Unformatted' = not formatted. Unformatted data has no record or field delimiters, and is entirely described by the specified Teradata PT schema.</li> </ul>
HadoopBlockSize= ( <i>x * 1K bytes</i> )	Optional attribute that specifies the size of the block /buffer, in 1K increments, when writing Hadoop/ HDFS files. The HadoopBlockSize value can be defined anywhere from 1 to <i>x</i> 1K bytes, where <i>x</i> is arbitrary. The typical default Hadoop/HDFS Cluster Block Size is 64MB which is also what TPT uses: (65536 * 1024 = 64MB).

Syntax Element	Description
	<p><b>Note:</b></p> <p>Before using this attribute to change the default, consult your system administrator. This value affects memory consumption (internal buffer allocated at runtime is twice this size), and should not be changed indiscriminately.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>• 1 - 2147483647</li> <li>• 0 = Default value</li> </ul> <p><b>Note:</b></p> <p>Default value = 65536.</p>
HadoopFileFormat= ' <i>hadoopFileFormat</i> '	<p>Optional attribute that specifies the format of the file that the TDCH job should process. For more information about the DataConnector's Hadoop interfaces and the <a href="#">Teradata Connector for Hadoop tutorial</a> for supported and default values, see <a href="#">Processing Hadoop Files and Tables</a>.</p>
HadoopHost= ' <i>hadoopHostName</i> '	<p>Optional attribute that specifies the host name or IP address of the NameNode in a Hadoop cluster. When launching a TDCH job, this value should be the host name or IP address of the node in the Hadoop cluster on which the TPT job is being run. This host name or IP address should be reachable by all DataNodes in the Hadoop cluster. For more information about the DataConnector's Hadoop interfaces. see <a href="#">Processing Hadoop Files and Tables</a>.</p> <p>When launching a HDFS API job this value indicates the cluster where the HDFS operation will be performed and can be set as follows:</p> <ul style="list-style-type: none"> <li>• "default" = The default name-node declared in the Hadoop HDFS configuration file.</li> <li>• &lt;host-name&gt;:&lt;port&gt; = The host-name/ip-address and port of the name-node on the cluster where the HDFS operation is to be performed. The "&lt;port&gt;" value is optional.</li> </ul>
HadoopJobType= ' <i>hadoopJobType</i> '	<p>Optional attribute that specifies the type of TDCH job to launch. For more information about the DataConnector's Hadoop interfaces and the <a href="#">Teradata Connector for Hadoop tutorial</a> for supported and default values, see <a href="#">Processing Hadoop Files and Tables</a>.</p>
HadoopNumMappers= ' <i>hadoopNumMappers</i> '	<p>Optional attribute that specifies the number of mappers that the TDCH will launch. For more information about the DataConnector's Hadoop interfaces and the <a href="#">Teradata Connector for Hadoop tutorial</a> for supported and default values, see <a href="#">Processing Hadoop Files and Tables</a>.</p>

Syntax Element	Description
	<a href="#">tutorial</a> for supported and default values, see <a href="#">Processing Hadoop Files and Tables</a> .
HadoopProperties = 'hadoopProperties'	Optional attribute that specifies one or more Hadoop properties and their values to be used by TPT when submitting the Hadoop command. For more information about the DataConnector's Hadoop interfaces and the <a href="#">Teradata Connector for Hadoop tutorial</a> for supported and default values, see <a href="#">Processing Hadoop Files and Tables</a> .
HadoopSeparator= 'hadoopSeparator'	Optional attribute that specifies the characters that separate fields in the file processed by the TDCH job. This attribute is only valid when 'HadoopFileFormat' is set to 'textfile', which is the attribute's default value. For more information about the DataConnector's Hadoop interfaces and the <a href="#">Teradata Connector for Hadoop tutorial</a> for supported and default values, see <a href="#">Processing Hadoop Files and Tables</a> .
HadoopSourceDatabase='hadoopSourceDatabase'	Optional attribute that specifies the name of the source database in Hive or Hcatalog from which data is exported. For more information about the DataConnector's Hadoop interfaces and the <a href="#">Teradata Connector for Hadoop tutorial</a> for supported and default values, see <a href="#">Processing Hadoop Files and Tables</a> .
HadoopSourceFieldNames = 'hadoopSourceFieldNames'	Optional attribute that specifies the names of the fields to export from the source HDFS files, or from the source Hive and HCatalog tables, in comma separated format. For more information about the DataConnector's Hadoop interfaces and the <a href="#">Teradata Connector for Hadoop tutorial</a> for supported and default values, see <a href="#">Processing Hadoop Files and Tables</a> .
HadoopSourcePartitionSchema= 'hadoopSourcePartitionSchema'	Optional attribute that specifies the full partition schema of the source table in Hive, in comma separated format. This attribute is only valid when 'HadoopJobType' is set to 'hive'. For more information about the DataConnector's Hadoop interfaces and the <a href="#">Teradata Connector for Hadoop tutorial</a> for supported and default values, see <a href="#">Processing Hadoop Files and Tables</a> .
HadoopSourcePaths= 'hadoopSourcePaths'	Optional attribute that specifies the directory of the to-be-exported source files in HDFS. This attribute is required when 'HadoopJobType' is set to 'hdfs', optional when 'HadoopJobType' is set to 'hive', and invalid when 'HadoopJobType' is set to 'hcat'. For more information about the DataConnector's Hadoop interfaces and the <a href="#">Teradata Connector for Hadoop tutorial</a> for supported and default values, see <a href="#">Processing Hadoop Files and Tables</a> .

Syntax Element	Description
	<a href="#">Hadoop tutorial</a> for supported and default values, see <a href="#">Processing Hadoop Files and Tables</a> .
HadoopSourceTable = ' <i>hadoopSourceTable</i> '	Optional attribute that specifies the name of the source table in Hive or Hcatalog from which data is exported. This attribute is required when 'HadoopJobType' is set to 'hcat', optional when 'HadoopJobType' is set to 'hive', and invalid when 'HadoopJobType' is set to 'hdfs'. For more information about the DataConnector's Hadoop interfaces and the <a href="#">Teradata Connector for Hadoop tutorial</a> for supported and default values, see <a href="#">Processing Hadoop Files and Tables</a> .
HadoopSourceTableSchema= ' <i>hadoopSourceTableSchema</i> '	Optional attribute that specifies the full-column schema of the source table in Hive or Hcatalog, in comma separated format. For more information about the DataConnector's Hadoop interfaces and the <a href="#">Teradata Connector for Hadoop tutorial</a> for supported and default values, see <a href="#">Processing Hadoop Files and Tables</a> .
HadoopTargetDatabase= ' <i>hadoopTargetDatabase</i> '	Optional attribute that specifies the name of the Target database in Hive or Hcatalog to which data is imported. It is optional with a 'hive' or 'hcat' job and not valid with an 'hdfs' job. For more information about the DataConnector's Hadoop interfaces and the <a href="#">Teradata Connector for Hadoop tutorial</a> for supported and default values, see <a href="#">Processing Hadoop Files and Tables</a> .
HadoopTargetFieldNames = ' <i>hadoopTargetFieldNames</i> '	Optional attribute that specifies the names of the fields to write to the target file in HDFS, or to the target Hive and HCatalog table, in comma separated format. For more information about the DataConnector's Hadoop interfaces and the <a href="#">Teradata Connector for Hadoop tutorial</a> for supported and default values, see <a href="#">Processing Hadoop Files and Tables</a> .
HadoopTargetPartitionSchema= ' <i>hadoopTargetPartitionSchema</i> '	Optional attribute that specifies the names of the fields to write to the target file in HDFS, or to the target Hive and HCatalog table, in comma separated format. For more information about the DataConnector's Hadoop interfaces and the <a href="#">Teradata Connector for Hadoop tutorial</a> for supported and default values, see <a href="#">Processing Hadoop Files and Tables</a> .
HadoopTargetPaths= ' <i>hadoopTargetPaths</i> '	Optional attribute that specifies the directory of the to-be-imported source files in HDFS. This attribute is required when 'HadoopJobType' is set to 'hdfs', optional when 'HadoopJobType' is set to 'hive', and invalid when 'HadoopJobType' is set to 'hcat'. For more information about the DataConnector's

Syntax Element	Description
	Hadoop interfaces and the <a href="#">Teradata Connector for Hadoop tutorial</a> for supported and default values, see <a href="#">Processing Hadoop Files and Tables</a> .
HadoopTargetTable= ' <i>hadoopTargetTable</i> '	Optional attribute that specifies the name of the target table in Hive or Hcatalog where data will be imported. This attribute is required when 'HadoopJobType' is set to 'hcat', optional when 'HadoopJobType' is set to 'hive', and invalid when 'HadoopJobType' is set to 'hdfs'. For more information about the DataConnector's Hadoop interfaces and the <a href="#">Teradata Connector for Hadoop tutorial</a> for supported and default values, see <a href="#">Processing Hadoop Files and Tables</a> .
HadoopTargetTableSchema= ' <i>hadoopTargetTableSchema</i> '	Optional attribute that specifies the full column schema of the target table in Hive or Hcatalog, in comma separated format. For more information about the DataConnector's Hadoop interfaces and the <a href="#">Teradata Connector for Hadoop tutorial</a> for supported and default values, <a href="#">Processing Hadoop Files and Tables</a> .
HadoopUser= ' <i>hadoopUser</i> '	Optional attribute that specifies the name of the Hadoop user to utilize when reading and writing files via the HDFSAPI interface. The currently logged-in user-name where the TPT HDFS job is running is used when this attribute is not specified. For more information about the DataConnector's Hadoop interfaces, see <a href="#">Processing Hadoop Files and Tables</a> .
IndicatorMode = ' <i>mode</i> '	Optional attribute that specifies whether indicator bytes are inserted at the beginning of each record. This attribute is used by both the producer and consumer operators. Valid values are: <ul style="list-style-type: none"><li>• 'Y[es]' = Indicator mode data. This value is not valid for the 'text' or 'delimited' record formats.</li><li>• 'N[o]' = Nonindicator mode data (default).</li></ul>
LobDirectoryPath = ' <i>pathName</i> '	This optional attribute defines an existing location where the producer operator will store temporary LOB files. This directory will only be used if DataConnector needs to transfer inline LOBs in deferred mode because all the inline LOBs cannot fit within a single request message. Teradata PT will automatically handle the creation and removal of the temporary LOB files. The user needs to provide the location.  If <i>pathName</i> is not provided, the current working directory will be used.  This attribute is not valid for the consumer operator and is not valid on the z/OS platform.

Syntax Element	Description
MultipleReaders = ' <i>option</i> '	<p>Optional attribute that, when set to 'Yes', instructs the DataConnector producer operator that more than one instance can be used to read a single file in parallel.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>'Y[es]' = Allow multiple producer instances to read a single file. <i>FileList</i> must not be set to 'Yes'.</li> <li>'N[o]' = Only the first instance of the producer operator is allowed to read the source data.</li> </ul> <p>This attribute is not valid for the consumer operator.</p> <p><b>Note:</b></p> <p><i>MultipleReaders</i> is disabled when the job has 1 or more inline LOB columns in the schema and the total schema size is greater than 1,024,000 bytes. A warning message will be displayed and the job will continue.</p>
NamedPipeTimeOut = <i>seconds</i>	<p>Optional attribute that enables checking of named pipes (FIFOs) by the producer operator. If <i>seconds</i> is set to a positive number, the DataConnector producer operator will check the pipe for data every second until either data becomes available, or the amount of time specified is reached and the job terminates. If the attribute is not specified, no checking of pipes will be performed. This will yield faster performance, but may also result in a hung job if data is not available in the pipe when it is read.</p> <p>This attribute is only for jobs that use the DataConnector producer operator to read pipes directly. It is not used when the Named Pipe Access Module (NPAM) performs the pipe I/O.</p> <p>This attribute is not valid for the consumer operator.</p>
NotifyExit = ' <i>inmodName</i> '	<p>Optional attribute that specifies the name of the user-defined notify exit routine with an entry point named _dynamn. If no value is supplied, the following default name is used:</p> <ul style="list-style-type: none"> <li>libnotifyext.dll for Windows platforms</li> <li>libnotifyext.dylib for Apple macOS platform</li> <li>libnotifyext.so for all other UNIX platforms</li> <li>NOTFYEXT for z/OS platforms</li> </ul> <p>NotifyMethod must be set to 'Exit'.</p> <p>This attribute is valid for both producer and consumer operators.</p> <p>See <a href="#">Deprecated Syntax</a> for information about providing your own notify exit routine.</p>
NotifyLevel = ' <i>notifyLevel</i> '	Optional attribute that specifies the level at which certain events are reported.

Syntax Element	Description
	<p>See <a href="#">DataConnector Operator Events</a> for a complete description of possible events and the notification level for each event. Valid values are:</p> <ul style="list-style-type: none"> <li>'Off' = No notification of events is provided (default).</li> <li>'Low' = Low Notification Level.</li> <li>'Med' = Medium Notification Level.</li> <li>'High' = High Notification Level.</li> </ul> <p>This attribute is valid for both producer and consumer operators.</p>
NotifyMethod = ' <i>notifyMethod</i> '	<p>Optional attribute that specifies the method for reporting events. The methods are:</p> <ul style="list-style-type: none"> <li>'None' = No event logging is done (default).</li> <li>'Msg' = Writes the events to a log.</li> <li>'Exit' = Sends the events to a user-defined notify exit routine.</li> </ul> <p>This attribute is valid for both producer and consumer operators.</p>
NotifyString = ' <i>notifyString</i> '	<p>Optional attribute that specifies a user-defined string to precede all messages sent to the system log. This string is also sent to the user-defined notify exit routine. The maximum length of the string is:</p> <ul style="list-style-type: none"> <li>80 bytes, if NotifyMethod is 'Exit'</li> <li>16 bytes, if NotifyMethod is 'Msg'</li> </ul> <p>This attribute is valid for both producer and consumer operators.</p>
NullColumns = ' <i>option</i> '	<p>Optional attribute that determines how missing columns are treated. This only applies to delimited data for the producer operator.</p> <p>To utilize this attribute, AcceptMissingColumns must be 'Y[es]' or 'Y[eswithoutlog]' and QuotedData must be 'Y[es]' or 'O[ptional]'.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>'Y[es]' = New job-created columns will be NULL (default).</li> <li>'N[o]' = New job-created columns will contain the empty string "".</li> </ul> <p>For the following examples, the delimiter character is the default   character, QuotedData is enabled and AcceptMissingColumns is 'Y'. The example schema is:</p> <pre style="background-color: #f0f0f0; padding: 10px;"> ... (VARCHAR(5), VARCHAR(5), VARCHAR(5), VARCHAR(5), VARCHAR(5), VARCHAR(5)) ... </pre>

Syntax Element	Description
	<p>The first example data record is:</p> <pre>"abc"   ""   "def"</pre> <p>The schema requires 6 fields but the record only provides 4.</p> <p>Fields 1, 2, and 4 contain the strings "abc", "", and "def".</p> <p>Note that "" is not NULL. Rather, it is a character string of zero length. It is handled in the same manner as any other string.</p> <p>Field 3 is an explicitly provided NULL column. Because it is part of the original record, it is not affected by the NullColumns attribute.</p> <p>Fields 5 and 6 are not provided and must be created by the DataConnector producer operator.</p> <p>If NullColumns is set to 'Y[es]', or the default behavior is used, the result will be as if the data file contained the record</p> <pre>"abc"   ""   "def"      </pre> <p>where both newly created columns are NULL.</p> <p>But if NullColumns = 'N[o]' is used, the behavior will be as if the record was defined as</p> <pre>"abc"   ""   "def"   ""   "</pre> <p>where the newly created columns contain empty strings.</p> <p>Note that fields 2 and 3, which were both part of the original data record, are unchanged regardless of the NullColumns attribute setting.</p> <p>This attribute is ignored by the consumer operator.</p>
OpenMode = ' <i>mode</i> '	<p>Optional attribute that specifies the read/write access mode.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>• 'Read' = Read-only access. Only valid for the producer operator.</li> <li>• 'Write' = Write-only access. Only valid for the consumer operator.</li> <li>• 'WriteAppend' = Write-only access appending to existing file. Only valid for the consumer operator.</li> </ul> <p>If <i>mode</i> is not specified for OpenMode, it defaults to 'Read' for the producer operator and 'Write' for the consumer operator.</p>
OpenQuoteMark = ' <i>character(s)</i> '	Optional attribute that allows you to define the character sequence for the opening quote mark

Syntax Element	Description
	<p>within delimited data. The default value is the double quote character, "".</p> <p>May be any single or multibyte value from the session character set. For example, ' ' or '  '.</p> <p>For the producer operator, QuotedData must be set to 'Yes' or 'Optional'. For the consumer operator, QuotedData must be set to 'Yes'.</p>
PrivateLogName = ' <i>logName</i> '	<p>Optional attribute that specifies the name of a log that is maintained by the Teradata PT Logger inside the public log. The private log contains all the diagnostic trace messages produced by the operator using the TraceLevel attribute.</p> <p>A hyphen and the instance number are appended to the <i>logName</i> value. For example, if PrivateLogName = 'DClog', then the actual log name for instance 1 is DClog-1. Similarly, for instance 2, it is DClog-2, and so on.</p> <p>The private log can be viewed using the tlogview command as follows, where jobId is the Teradata PT job name and privatelogname is the value for the operator's PrivateLogName attribute, including hyphen and instance number:</p> <pre>tlogview -j jobId -f privatelogname</pre> <p>If the private log is not specified, all output is stored in the public log.</p> <p>For more information about the tlogview command, see <a href="#">Teradata PT Utility Commands</a>.</p> <p>This attribute is valid for both producer and consumer operators.</p>
QuotedData = ' <i>option</i> '	<p>Optional attribute that determines if delimited data is expected to be enclosed within quotation marks.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>• 'Y[es]' = All columns are expected to be enclosed in quotation marks. For a producer operator, quotation marks will be discarded before data is written to the data stream. For a consumer operator, quotation marks are added before column data is written to the output file.</li> <li>• 'N[o]' = Column enclosure within quotation marks is not expected (default). For a producer operator, quotation marks are treated as data. For a consumer operator, quotation marks will not be added to column data.</li> <li>• 'Optional' = Columns can optionally be enclosed within quotation marks. For a producer operator, if quotation marks are found, they will be discarded before data is written to the data stream. For a consumer operator, quotation</li> </ul>

Syntax Element	Description
	marks are only added if column data contains an embedded quotation mark or an embedded text delimiter.
RecordErrorFileName = 'filePath'	<p>Optional attribute that specifies where error records are stored when reading delimited data files. Error records include those with either incorrect column counts or columns with invalid lengths. This only applies to the producer operator.</p> <p>The ErrorLimit attribute specifies how many error records can be written to this file.</p> <p>If this attribute is undefined, the first occurrence of an error record will result in a fatal operator error and job termination.</p> <p>This attribute is ignored by the consumer operator.</p>
RecordErrorVerbosity = 'option'	<p>Optional attribute that allows for annotations in the record error file when the RecordErrorFileName attribute is set. This only applies to the producer operator.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>• 'Off' = No annotations are to be inserted into the record error file (default).</li> <li>• 'Low' = The error message describing the nature of the error is included.</li> <li>• 'Med' = The file name and record number are included, along with error messages describing the nature of the error.</li> <li>• 'High' = The same as 'Med'.</li> </ul> <p>This attribute is ignored by the consumer operator.</p>
RecordsPerBuffer = count	<p>Optional attribute that defines the number of records to be processed by each instance of the producer operator during each processing phase when MultipleReaders='Yes'. Valid values are between 1 and 999999.</p> <p>The default is calculated by dividing the buffer size (1048575) by the number of worker reader instances.</p> <p>That result is then divided by the maximum record size as defined by the schema.</p> <p>The number of worker instances is equal to the total operator instances minus 1.</p> <p>For example, if 10 reader instances are defined, and the length of the schema is 400 bytes, then this value would default to 1048575 bytes / 9 instances / 400 bytes = 291 records.</p> <p>This attribute is ignored by the consumer operator.</p>

Syntax Element	Description
RowsPerInstance = <i>rows</i>	<p>Optional attribute that specifies the maximum number of records to be processed by each instance of the producer operator.</p> <p>This number spans files, meaning that processing continues over multiple files until the row limit is reached for each instance. Once the row limit is reached for a particular instance, that instance is done. If the limit is not reached for a particular instance, that instance ends normally.</p> <p>The limit is not effective across restarts, meaning the row count is reset to zero upon restart.</p> <p>This attribute is ignored by the consumer operator.</p>
SingleRecordWriting = ' <i>option</i> '	<p>Optional attribute that allows an Access Module to receive one record per write operation instead of receiving a buffer that can contain multiple records. This option only applies to consumer instances that have a valid AccessModuleName entry.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>'Y[es]' = Each write operation to the Access Module will only contain one record.</li> <li>'N[o]' = Each write operation to the Access Module will contain a buffer that may include multiple records. 'No' is the default value.</li> </ul> <p>This attribute is ignored by the producer operator.</p>
SkipRows = <i>rows</i>	<p>Optional attribute that specifies the number of rows to skip by each instance of the producer operator. The SkipRowsEveryFile attribute will determine if SkipRows spans files and restarts.</p> <p>This attribute is not valid for the consumer operator.</p>
SkipRowsEveryFile = ' <i>option</i> '	<p>Optional attribute that governs the behavior of SkipRows. This only applies to the producer operator.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>'Y[es]' = SkipRows restarts at the beginning of each file. For example, if SkipRows = 5, SkipRowsEveryFile = 'Yes', and 5 files to be processed each contain 300 rows, the first 5 rows of each file are skipped and rows 6 through 300 are processed. You might use this option to skip repetitive header rows in each file to be processed.</li> <li>'N[o]' = SkipRows value is cumulative (default). That is, processing continues over multiple files until the specified number of rows to skip is reached. For example, if SkipRows = 1000, SkipRowsEveryFile = 'N', and 5 files to be processed each contain 300 rows, Files 1, 2, and 3 are skipped in their entirety, file 4 begins processing at row 101, and all file 5 is processed.</li> </ul>

Syntax Element	Description
	<p>You might use this option to skip rows that were already processed in a failed job.</p> <p>This attribute is ignored by the consumer operator.</p>
TextDelimiter = ' <i>character</i> '	<p>Optional attribute that specifies the bytes that separate fields in delimited records. Any number of single or multibyte characters from the session character set can be defined. This attribute is only valid when Format = 'Delimited'.</p> <p>The default delimiter value is the pipe character (   ).</p> <p>The TextDelimiter byte sequence can be treated as data if it is preceded with an escape sequence defined by the EscapeTextDelimiter attribute.</p> <p><b>Note:</b></p> <p>To use the tab character as the delimiter character, specify TextDelimiter = 'TAB'. Use uppercase "TAB" not lowercase "tab".</p> <p>This attribute is used by both the producer operator and the consumer operator.</p>
Timeout = <i>seconds</i>	<p>Optional attribute that specifies the number of seconds the system waits for input to finish by the producer operator. The supplied value is passed to all access modules attached to the producer operator.</p> <p>Valid values are from 1 to 99999 seconds.</p> <p>The default is 0 (no timeout).</p> <p>This attribute is not valid for the consumer operator.</p>
TraceLevel = ' <i>level</i> '	<p>Optional attribute that specifies the types of diagnostic information that are written by each instance of the operator to the public log (or private log, if one is specified using the PrivateLogName attribute).</p> <p>The diagnostic trace function provides detailed information in the log file to aid in problem tracking and diagnosis. The trace levels are:</p> <ul style="list-style-type: none"> <li>• 'None' = Enables minimal information such as initialization, access module attach/detach operations, file opening, error messages, and statistics (number of records processed and processor time used). This is the default.</li> </ul> <p>The PrivateLogFile attribute default is used only if a TraceLevel attribute other than 'None' is specified. If a TraceLevel attribute other than 'None' is specified without a PrivateLog specification, the DataConnector operator generates a private log name and a</p>

Syntax Element	Description
	<p>message containing the private log name is issued in the public log.</p> <p>If no TraceLevel attribute is specified, or if the specified value is 'None', and the PrivateLogFile attribute is specified, the TraceLevel is set to 'Milestones'. The recommended TraceLevel value is 'None', which produces NO log file. Specifying any value greater than 'IO_Counts' produces a very large amount of diagnostic information.</p> <ul style="list-style-type: none"> <li>• 'Milestones' = Enables the trace function only for major events such as initialization, access module attach/detach operations, file openings and closings, error conditions, and so on</li> <li>• 'IO_Counts' = Enables the trace function for major events and I/O counts</li> <li>• 'IO_Buffers' = Enables the trace function for major events, I/O counts, and I/O buffers</li> <li>• 'All' = Enables the trace function for major events and I/O counts and I/O buffers plus function entries.</li> </ul> <p>If PrivateLogFile attribute specifies a log file without specifying the TraceLevel attribute, "minimal" statistics are displayed in the log file, such as:</p> <ul style="list-style-type: none"> <li>• Name of files as they are processed</li> <li>• Notice when sending rows begin</li> <li>• On completion, the number of rows processed and the CPU time consumed</li> <li>• Total files processed and CPU time consumed by each instance of the DataConnector operator</li> </ul> <p><b>Note:</b></p> <p>The TraceLevel attribute is provided as a diagnostic aid only. The amount and type of additional information provided by this attribute will change to meet evolving needs from release to release. It should be used with caution, due to the impact it can have on performance and disk usage.</p> <p>This attribute is valid for both producer and consumer operators.</p>
TrimChar = ' <i>character</i> '	<p>Optional attribute that specifies the characters to be trimmed from delimited column data by the producer operator. Use in conjunction with the TrimColumns attribute.</p> <p>Rules for a trim character are:</p> <ul style="list-style-type: none"> <li>• The trim character must be a single character, but may be either a single-byte or multi-byte character from the session character set.</li> <li>• The default value is the blank (space) character.</li> </ul>

Syntax Element	Description
	<ul style="list-style-type: none"> <li>Trimming can be performed on either quoted or unquoted fields.</li> <li>If a field consists solely of one or more trim characters, it will be a zerolength VARCHAR after trimming.</li> </ul> <p>For example, if TextDelimiter = ' ', TrimChar = 'a', and TrimColumns = 'Both', then the following delimited record:</p> <pre>a1 2aa aaa aaaa4aaaa</pre> <p>will get treated as:</p> <pre>1 2  4</pre> <p>This attribute is ignored by the consumer operator.</p>
TrimColumns = ' <i>option</i> '	<p>Optional attribute that specifies how characters are trimmed from delimited column data by the producer operator. Use in conjunction with the TrimChar attribute.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>'None' = No trimming (default)</li> <li>'Leading' = Leading characters are trimmed</li> <li>'Trailing' = Trailing characters are trimmed</li> <li>'Both' = Both leading and trailing characters are trimmed</li> </ul> <p><b>Note:</b> If TrimColumns and TruncateColumnData are enabled, trimming occurs before truncating.</p> <p>This attribute is not valid for the consumer operator.</p>
TruncateColumnData = ' <i>option</i> '	<p>Optional attribute that determines how a column is treated whose length is greater than that defined in the schema. This only applies to delimited data for the producer operator.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>'Y[es]' = The column is truncated to the maximum length and processed without an error being raised. The edited record is sent to the database and the original record is saved in the record error file, if it is defined via the RecordErrorFileName attribute.</li> <li>'N[o]' = A column that is too long is not sent to the database, but the original record is saved in the record error file, if it is defined via the RecordErrorFileName attribute. If RecordErrorFileName is not defined, a fatal error will occur. 'No' is the default value.</li> </ul>

Syntax Element	Description
	<ul style="list-style-type: none"> <li>'YesW[ithoutLog]' = The column is truncated to the maximum length and processed without an error being raised. The edited row is sent to the database, but the original record is not saved in the record error file.</li> </ul> <p>This attribute is ignored by the consumer operator.</p>
VigilElapsedTime = <i>minutes</i>	<p>Optional attribute that specifies the elapsed time from the beginning of the job to the end of the job for an Active Directory Scan by the producer operator. This is the amount of time to wait from the VigilStartTime in which the directory specified in the DirectoryPath attribute is watched for the arrival of new files. If the VigilStartTime attribute is not set, the system's current time is used as the start time. VigilElapsedTime is expressed in minutes. For example, a 2-hour and 15-minute window is indicated as:</p> <pre>VigilElapsedTime = 135</pre> <p>VigilElapsedTime and VigilStopTime are interchangeable, but mutually exclusive. There is no default value for VigilElapsedTime. It requires the ArchiveDirectoryPath to also be set.</p> <p>This attribute is not valid for the consumer operator.</p>
VigilMaxFiles = <i>numberOfFiles</i>	<p>Optional attribute that defines the maximum number of files that can be scanned in one pass by the producer operator for an Active Directory Scan. Greater values require more Teradata PT global memory and could degrade performance.</p> <p>The valid value range of <i>numberOfFiles</i> is from 10 to 50000. The default value is 2000.</p> <p>Use of the VigilMaxFiles attribute requires that a value for the ArchiveDirectoryPath attribute be specified, along with VigilStartTime and VigilStopTime (or VigilElapsedTime).</p> <p>The attribute's value can be modified during job execution using the External Command Interface. To change the value of VigilMaxFiles during execution, enter:</p> <pre>twbcmd &lt;Teradata PT job ID&gt; &lt;operator ID&gt; VigilMaxFiles=&lt;number of files&gt;</pre> <p>This attribute is ignored by the consumer operator.</p>
VigilNoticeFileName = ' <i>noticeFileName</i> '	Optional attribute that specifies the name of the file in which the vigil notice flag is to be written by the producer operator for an Active Directory Scan.

Syntax Element	Description
	<p>For example, to request that a record be written to the file /home/user/Alert.txt, specify the attribute as:</p> <pre>VigilNoticeFileName = '/home/user/ Alert.txt'</pre> <p>If a directory path is not specified, the file is saved in the current working directory.</p> <p>Naming a file activates this notification feature. It requires that a value for the ArchiveDirectoryPath attribute be specified, along with VigilStartTime and VigilStopTime (or VigilElapsedTime). This attribute is ignored by the consumer operator.</p>
VigilSortField = ' <i>sortTime</i> '	<p>Optional attribute that provides the capability for files to be sorted in a specific order by the producer operator for an Active Directory Scan.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>• 'TIME' = All files will be sorted according to the time they were last modified.</li> <li>• 'NAME' = All files are sorted by filename and processed in ascending alphabetical order.</li> <li>• 'NONE' = The sort feature is off (default).</li> </ul> <p>Since times associated with the files are tracked to the nearest second, more than one file may have the same timestamp. When modification times for files are less than one second apart, the sort order of the files may not represent the actual order modified.</p> <p>When using multiple instances, files cannot be processed in a specific sorted order. When this attribute is used, Teradata PT allows only a single instance of the DataConnector producer operator to be used in a job step. If more than one instance is specified, the job will fail.</p> <p>It requires that a value for the ArchiveDirectoryPath attribute be specified, along with VigilStartTime and VigilStopTime (or VigilElapsedTime).</p> <p>This attribute is not valid for the consumer operator.</p> <p><b>Note:</b> This attribute is not available for z/OS systems.</p>
VigilStartTime = 'yyyymmdd hh:mm:ss'	<p>Optional attribute that specifies the time to start the vigil time window for an Active Directory Scan by the producer operator. It helps define the period in which the directory specified in the DirectoryPath attribute is watched for the arrival of new files.</p> <p>The start time is expressed as follows:</p> <ul style="list-style-type: none"> <li>• yyyy is the 4-digit year (2000-3000)</li> </ul>

Syntax Element	Description
VigilStartTime = 'yyyymmdd hh:mm:ss'	<ul style="list-style-type: none"> <li>• mm is the month (1-12)</li> <li>• dd is the day of the month (1-31)</li> <li>• hh is the hour of the day (0-23)</li> <li>• mm is the minute (0-59)</li> <li>• ss is the second (0-59)</li> </ul> <p>For example, a start time of August 23, 2019, at 9:22:56 a.m. becomes:</p> <pre>VigilStartTime = '20190823 09:22:56'</pre> <p>This attribute is required for the VigilWaitTime attribute to work.</p> <p>There is no default value for VigilStartTime. It requires the ArchiveDirectoryPath to also be set, along with either VigilStopTime or VigilElapsedTime.</p> <p>This attribute is not valid for the consumer operator.</p>
VigilStopTime = 'yyyymmdd hh:mm:ss'	<p>Optional attribute that specifies the time to stop the vigil time window for an Active Directory Scan by the producer operator. It helps define the period in which the directory specified in the DirectoryPath is watched for the arrival of new files.</p> <p>The stop time is expressed as follows:</p> <ul style="list-style-type: none"> <li>• yyyy is the 4-digit year (2000-3000)</li> <li>• mm is the month (1-12)</li> <li>• dd is the day of the month (1-31)</li> <li>• hh is the hour of the day (0-23)</li> <li>• mm is the minute (0-59)</li> <li>• ss is the second (0-59)</li> </ul> <p>For example, a stop time of August 23, 2019, at 2 p.m. becomes:</p> <pre>VigilStopTime = '20190823 14:00:00'</pre> <p>VigilStopTime and VigilElapsedTime are interchangeable, but mutually exclusive. There is no default value for VigilStopTime. It requires the ArchiveDirectoryPath to also be set, along with VigilStartTime.</p> <p>This attribute is not valid for the consumer operator.</p>
VigilWaitTime = <i>waitSeconds</i>	<p>Optional attribute that specifies the amount of time to wait before starting to check the directory again if no new files were found by the producer operator for an Active Directory Scan.</p> <p>A wait time of 2 minutes becomes:</p> <pre>VigilWaitTime = 120</pre>

Syntax Element	Description
	<p>The default wait time is 60 seconds.</p> <p>Use of the VigilWaitTime attribute requires that a value for the ArchiveDirectoryPath attribute be specified, along with VigilStartTime and VigilStopTime (or VigilElapsedTIme).</p> <p>The attribute's value can be modified during job execution using the External Command Interface. To change the value of VigilWaitTime during execution, enter:</p> <pre>twbcmd &lt;Teradata PT job ID&gt; &lt;operator ID&gt; VigilWaitTime=&lt;Seconds&gt;</pre> <p>This attribute is ignored by the consumer operator.</p>
WriteBOM = ' <i>option</i> '	<p>Optional attribute that allows a BOM (Byte Order Mark) to be inserted at the beginning of a Unicode output file.</p> <p>This option only applies to consumer instances that are writing to text files (the Format attribute is 'Text' or 'Delimited') which are using a UTF-8 or UTF-16 character set.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>'Y[es]' = Inserts a BOM to the beginning of the output file based on the character set encoding.</li> <li>'N[o]' = Does not insert a BOM to the beginning of a Unicode output file. 'No' is the default value.</li> </ul> <p>This attribute is not valid for the producer operator.</p>

## Usage Notes

### FileName

The use of the FileName attribute varies depending on operator type, operating system, and whether the file resides in the local filesystem or in Hadoop's distributed file system. The traditional DataConnector attributes, including FileName, are used when interfacing with Hadoop via the HDFS API interface, but are not used when interfacing with Hadoop through TDCH. For more information about the DataConnector's Hadoop interfaces, see [Processing Hadoop Files and Tables](#).

- **DataConnector Producer Operator**

When using the DataConnector operator as a producer to read data from files in the local file system, the wildcard character (\*) is allowed in a FileName attribute if you want to process all matching files or members within a named UNIX OS directory or the z/OS partitioned dataset (PDS or PDSE). Wildcard UNIX-style "egrep" patterns are also supported when using the DataConnector operator as a producer to read Hadoop files via the HDFS API interface.

The following conditions also apply depending on your operating system:

- On UNIX systems, the *FileName* attribute limit is 255 bytes. *FileName* can either be a complete pathname of the file, or the name of a file within a directory . But if the directory is not defined in the optional *DirectoryPath* attribute, *filename* is expected to be found in the default directory. See the table in this section for examples.
- On z/OS systems, *FileName* can specify the fully-qualified dataset name of the data file, including the member name if the dataset is a PDS or PDSE, the member name of a PDS or PDSE library, or 'DD:<ddname>'. If only a member name is specified for the *FileName*, then the (PDS or PDSE) dataset name containing the member must be specified by the *DirectoryPath* attribute. See the table in this section for examples.

#### • **DataConnector Consumer Operator**

When using the DataConnector operator as a consumer, the *FileName* attribute becomes the complete file specification, and the *FileName* cannot contain the wildcard character (\*).

On UNIX systems, unless you specify a pathname, the *FileName* is expected to be found in the default directory. See the table in this section for examples.

When writing files whose *FileName* value is not fully qualified into the Hadoop distributed file system via the HDFS API interface, the file will be created in the directory of the user specified by the *HadoopUser* attribute.

#### **Writing Files with Multiple Instances**

The Consumer Operator can write multiple files with multiple instances. When more than one instance is specified, the output will be spread across multiple files. The distribution of records in each file will not be uniform. If you want each file to have the same (approximate) number of records, use the -C command line option, which forces the rows to the consumer operator in a round-robin fashion.

There are two methods by which the output files can be specified:

- Use the *FileList* attribute and name each file explicitly in the *FileList* file.
- Allow the DataConnector Operator to generate the file names automatically.

When the file names are created automatically, the following procedures will be used:

#### **UNIX File Names**

File names will be created by adding a dash and a number starting with one and incremented by one. For example, a file-name of "aa" with two instances will create the following two files:

- aa-1
- aa-2

#### **ZIP and GZIP File Names**

When ZIP and GZIP files are created, the naming is slightly different from UNIX file names because the file-type extension must be preserved. Using the example with the file names "aa.gz"/"aa.zip" and two instances, the following two files will be created:

- aa-1.gz/aa-1.zip

- aa-2.gz/aa-2.zip

## Z/OS Datasets

The multiple writer feature is enabled on z/OS by generating sequential "dd-names." This feature also supports using a PDS/PDSE dataset syntax expression to generate sequential PDS member names.

There are two accepted formats:

- DD-NAME FORMAT: { DD:xxx }
- DATASET FORMAT: { //'dataset-name(member-name)' }

The dd-name (xxx) and the (member-name) will be used as templates. The dd-name or member name provided can be 0-8 characters.

When there are (0) characters the following will be generated in sequential order:

- "D0000001"
- "D0000002"
- Etc. ( ... )

The maximum number of decimal digits will always be generated with zero fill.

When the number of characters provided is greater than the number of digits required, the digits will expand into and replace (x) characters where x = required - available.

When the number of characters provided is less than the number of required digits, the decimal digits will be expanded with zero fill so that in all cases the dd-name or member name will occupy 8 characters.

### DD-NAME Examples

Zero Characters "DD:" with 2 instances:

- "DD:D0000001"
- "DD:D0000002"

Two Characters "DD:XX" with 10 Instances:

- "DD:XX000001"
- "DD:XX000002"
- "DD:XX000003"
- "DD:XX000004"
- "DD:XX000005"
- "DD:XX000006"
- "DD:XX000007"
- "DD:XX000008"
- "DD:XX000009"

- "DD:XX000010"

Eight Characters "DD:XXXXXXXX" with 5 Instances:

- "DD:XXXXXXX1"
- "DD:XXXXXXX2"
- "DD:XXXXXXX3"
- "DD:XXXXXXX4"
- "DD:XXXXXXX5"

Eight Characters "DD:XXXXXXXX" with 10 Instances:

- "DD:XXXXXX01"
- "DD:XXXXXX02"
- "DD:XXXXXX03"
- "DD:XXXXXX04"
- "DD:XXXXXX05"
- "DD:XXXXXX06"
- "DD:XXXXXX07"
- "DD:XXXXXX08"
- "DD:XXXXXX09"
- "DD:XXXXXX10"

#### Dataset Member-Name Examples

Zero Characters "://" with 2 instances:

- "://"
- "://"

Two Characters "://" with 10 Instances:

- "://"
- "://"
- "://"
- "://"
- "://"
- "://"
- "://"
- "://"
- "://"
- "://"

- "://"MYDSN(XX000010)"

Eight Characters "://"MYDSN(XXXXXXX)" with 5 Instances:

- "://"MYDSN(XXXXXXX1)"
- "://"MYDSN(XXXXXXX2)"
- "://"MYDSN(XXXXXXX3)"
- "://"MYDSN(XXXXXXX4)"
- "://"MYDSN(XXXXXXX5)"

Eight Characters "://"MYDSN(XXXXXXX)" with 10 Instances:

- "://"MYDSN(XXXXXX01)"
- "://"MYDSN(XXXXXX02)"
- "://"MYDSN(XXXXXX03)"
- "://"MYDSN(XXXXXX04)"
- "://"MYDSN(XXXXXX05)"
- "://"MYDSN(XXXXXX06)"
- "://"MYDSN(XXXXXX07)"
- "://"MYDSN(XXXXXX08)"
- "://"MYDSN(XXXXXX09)"
- "://"MYDSN(XXXXXX10)"

- **Combining FileName and FileList Attributes**

The FileList attribute extends the capabilities of the FileName attribute. Adding FileList = 'Y' indicates that the file identified by FileName contains a list of files to be processed as input or used as containers for output. The file names found within the FileName file are expected to be full path specifications. If no directory name is included, the files are expected to be located within the current directory.

Supplying full paths for output files enables you to write files to multiple directories or disks. You cannot use the DirectoryPath attribute in conjunction with this feature.

When the combination of FileName and FileList attributes are used to control output, the supplied file list must have the same number of files as there are defined consumer instances; a mismatch results in a terminal error. At execution, rows are distributed to the listed files in a round-robin fashion if the `tbuild -C` option is used. Without the option, rows may not be evenly distributed across the listed files.

**Note:**

DataConnector operator supports a `FileList` file encoded in ASCII on workstation-attached platforms and EBCDIC on mainframe-attached platforms.

You cannot combine this feature with the archiving feature. Any attempt to use the archive feature (for example, by defining the `ArchiveDirectoryPath` attribute) results in a terminal error.

- Limiting the Size of Output Files**

If the DataConnector consumer operator is used to write many records to disk, the `FileSizeMax` attribute can be used to write records across several more manageable sized files instead of one very large file. The `FileSizeMax` attribute limits output files to a user-specific size. Whenever the size limit is reached, the current output file is closed and the next set of records are written to a new output file.

If the pathname that you specify with the `FileName` attribute (as *filename*) contains any embedded pathname syntax (“/” on a UNIX OS or “\” on Windows), the pathname is accepted as the entire pathname. However, if the `DirectoryPath` attribute is present, the `FileName` attribute is ignored, and a warning message is issued.

If the `FileList` file-name does not exist in HDFS, then the Data Connector will assume it is a local file and process it accordingly, otherwise if it is an HDFS file it will be read from the HDFS file system.

The following table contains valid syntax examples for the `FileName` attribute.

Operating System	Valid Syntax	Explanation
z/OS	<code>FileName = //''name.name(member)'''</code>	<p>z/OS PDS DSN: Name.Name(Member) where:</p> <ul style="list-style-type: none"> <li>• Name.Name = dataset Name.Name</li> <li>• Member = PDS Member.</li> </ul>
	<code>FileName = '/''name.name'''</code>	<p>z/OS DSN (sequential): Name.Name where:</p> <ul style="list-style-type: none"> <li>• Name.Name = dataset Name.Name.</li> </ul>
	<code>FileName = 'DD:ddname'</code>	<p>z/OS DSN is described in the JCL DD statement name “ddname.” If no DD statement is specified, the following occurs:</p> <ul style="list-style-type: none"> <li>• For input, the <code>fopen</code> library function tries to open an HFS file named <code>DD:ddname</code> in the home directory. If the file is not found, the <code>fopen</code> library function returns an error that is displayed in the <code>SYSOUT</code>.</li> <li>• For output, the <code>fopen</code> library function tries to create an HFS file named <code>DD:ddname</code> in the home directory. If the file already exists, the previous contents are overwritten.</li> </ul>
	<code>FileName = 'member'</code>	z/OS PDS member is expected to reside in the DSN that is defined in the <code>DirectoryPath</code> attribute.

Operating System	Valid Syntax	Explanation
UNIX	FileName = '/tmp/ user/filename'	UNIX pathname.
	FileName = 'filename'	If the DirectoryPath attribute is undefined, <i>filename</i> is located in the default directory.
Windows	FileName = ' tmp\user-filename'	Windows path name.
	FileName = 'filename'	Windows file name expected to be found in the directory defined in the DirectoryPath attribute. If the DirectoryPath is not defined, <i>filename</i> is located in the default directory.

**Note:**

On Windows platforms, using the wildcard character (\*) in *filename* can inadvertently include undesired files. For example, specifying \*.dat is the same as specifying \* .dat\*, which can include files with extensions such as .data, .date, and .dat071503. Therefore, it is recommended that extraneous files be removed from your folder.

## Format

Several file formats can be processed by the DataConnector operator. Specify file format with the Format attribute.

- **Format = 'Binary'** – Each record contains a two-byte integer data length (*n*) followed by *n* bytes of data.
- **Format = 'Binary4'** – Each record contains a four-byte data integer data length (*n*) followed by *n* bytes of data.
- **Format = 'Text'** – Each record is entirely character data, an arbitrary number of bytes followed by one of the following end-of-record markers:
  - A single-byte line feed (X'0A') on UNIX platforms
  - A double-byte carriage-return/line-feed pair (X'0D0A') on Windows platforms
 See [End of Record Markers for Text Formatted Data](#).
- **Format = 'Delimited'** – Each record is in variable-length text record format, but each contains fields (columns) separated by one or more delimiter characters, as defined with the TextDelimiter attribute, which has the following limitations:
  - It can only be a sequence of characters.

- It cannot be any character that appears in the data.
- It cannot be a control character other than a tab.

With this file format, when using the DataConnector Operator as a producer, the operator's associated TPT schema object must comprise only VARCHAR, JSON, JSON BY NAME, CLOB BY NAME, BLOB BY NAME, XML BY NAME, XML, CLOB or VARDATE columns. And, if not provided, the TextDelimiter attribute defaults to the pipe character ( | ).

**Note:**

There is no default escape character when using delimited data. Use the DataConnector operator EscapeTextDelimiter attribute to define the escape character.

See [End of Record Markers for Text Formatted Data](#).

- **Format = 'Formatted'** – Each record is in a format known as FastLoad or Teradata format, which is a two-byte integer (*n*) followed by *n* bytes of data, followed by an end-of-record marker (X'0A' or X'0D0A').
- **Format = 'Formatted4'** – A version of the FastLoad format that supports rows greater than 64KB in length. Each record is a four-byte integer (*n*) followed by *n* bytes of data, followed by an end-of-record marker (X'0A') or X'0D0A').
- **Format = 'Unformatted'** – The data does not conform to any predefined format. Instead, the data is entirely described by the columns in the schema definition of the DataConnector operator.

**Note:**

When processing Hadoop files and tables via the TDCH interface, data is transferred between TPT and TDCH in formatted mode with indicator bytes; the value of the HadoopFileFormat attribute determines the format of the Hadoop file or table processed by TDCH.

## OpenMode

Attribute that specifies the read/write access mode. Read means read-only access. Write means write-only access. If a mode value is not specified for OpenMode, it defaults to Read for a producer instance and Write for a consumer instance.

When interfacing with Hadoop systems whose version is earlier than 2.0 via the HDFS API interface, the value 'WriteAppend' is not supported.

## TextDelimiter and EscapeTextDelimiter

The text delimiter separates column values in delimited data (also known as VARTEXT). Whenever the text delimiter is encountered within the data, it will be interpreted as the end of the current column.

- The default text delimiter is a single vertical pipe ('|').
- For comma-separated values (csv) files, the delimiter is a comma (',').

- Multibyte delimiters are allowed as well, such as double pipe ('| '|').

If, for example, the default delimiter the pipe ( | ) is used, the following record (abcd|ef|gh|ijk|lmno) is broken into the following five text formatted columns:

- abcd
- ef
- gh
- ijk
- lmno

In the following example, comma delimiters are used, but the second record (third column) uses a comma within the data ("Silver Customer, since 1999").

First Name	Last Name	Note
John,	Doe,	Gold Customer
Sam,	Smith,	Silver Customer, since 1999

To resolve the ambiguity in the use of the comma in the second record (third column), use one of the following solutions:

- Use multiple single byte characters as shown in the following table (the TextDelimiter being used is ',',').

First Name	Last Name	Note
John, ,	Doe, ,	Gold Customer
Sam, ,	Smith, ,	Silver Customer, since 1999

- Define an escape text delimiter character using the EscapeTextDelimiter attribute (for example, a backslash ['\']), as in the following example, to embed the comma in the second record (third column).

First Name	Last Name	Note
John,	Doe,	Gold Customer
Sam,	Smith,	Silver Customer\, since 1999

## Rules for Quoted Delimited Data Handling

### Rules for Quotes

The following rules apply to both the open and close quote:

- If the open quote and close quote are distinct from each other, neither can be a substring of the other.
- Neither open quote nor close quote can be a substring of the delimiter.

- The delimiter cannot be a substring of either open quote or close quote.
- The backslash character (\) cannot occur in either the open quote or close quote.
- The “end of record” (EOR) (i.e., LF or CR/LF) and the EscapeRecordDelimiter sequence are treated as data within quotes.

## Rules for Parsing

- If values are unquoted, scan for the delimiter or end-of-line, since these are the only significant characters.
- If values are always quoted, the following rules apply:
  - At the start of the input line, or after a delimiter, an open quote must be present; otherwise, it is a malformed input line.
  - Following an open quote, all characters become part of the data value, with the following exceptions:

**A doubled close quote** – causes one close quote to become part of the data value. The DataConnector operator supports this behavior automatically so that you do not need to specify a quote as the EscapeQuoteDelimiter attribute.

**An EscapeQuoteDelimiter followed by a close quote** – causes the EscapeQuoteDelimiter to be discarded and the close quote to become part of the data value.

**An EscapeQuoteDelimiter followed by an EscapeQuoteDelimiter** – causes the first EscapeQuoteDelimiter to be discarded and the second EscapeQuoteDelimiter to become part of the data value

**An EscapeQuoteDelimiter followed by an open quote** – causes the EscapeQuoteDelimiter to be discarded and the open quote to become part of the data value.

---

### Note:

Escaping open quotes is not required. Unescaped open quotes will be treated as part of the data value.

---

**An undoubled, unescaped close quote** – terminates the data value, and must be immediately followed by a delimiter or end-of-line.

- If values are optionally quoted, the following rules apply:
  - At the start of the input line, or after a delimiter, if an open quote is present, the value is quoted and the rules for always-quoted values apply.
  - Otherwise, the value is unquoted, and the mentioned rules for unquoted values apply.

## Processing Hadoop Files and Tables

In addition to reading and writing from flat files and access modules, the DataConnector operator also has the ability to read and write Hadoop files and tables. Based on the set of attributes submitted with the DataConnector operator, one of two Hadoop interfaces will be used.

### HDFS API Interface

- When the attribute HadoopHost is specified with the DataConnector operator, the operator will use the HDFS API interface to process Hadoop files.
- The TPT HDFS Interface can be invoked on the cluster or on a remote Hadoop Client. The HadoopHost property identifies the cluster where the HDFS operation is to be performed and its presence activates the HDFS operation. All standard file functions and features of the Data Connector will be performed on the HDFS file system when it has been activated.

### TDCH-TPT Interface

When any Hadoop attributes besides HadoopHost and HadoopUser are submitted with the DataConnector operator, the operator will use the TDCH-TPT interface to process Hadoop files and tables. Throughout the rest of this section, these Hadoop attributes will be referred to as TDCH-specific Hadoop attributes.

#### Teradata Connector for Hadoop

The Teradata Connector for Hadoop, or TDCH, is a set of APIs and tools that supports high-performance parallel bi-directional data movement between Teradata systems and products in the Hadoop ecosystem. TDCH is built atop the MapReduce framework, and uses its distributed nature to offer extreme scalability and performance when transferring data between Teradata systems and Hadoop. For more information about TDCH, see the [Teradata Connector for Hadoop tutorial](#).

#### Utilizing TDCH in TPT Scripts via the TDCH-TPT Interface

- The TDCH-TPT interface is a bridge between TPT and TDCH. The TDCH-TPT interface extends TDCH to support Hadoop file and table transfers to TPT, and vice versa. This interface allows TPT users to use all the pre-existing TDCH functionality within a TPT script, and also allows to use TPT-specific functionalities alongside TDCH.
- When a TPT job script includes the DataConnector operator alongside any of the TDCH-specific Hadoop attributes, the DataConnector operator will launch a TDCH job using those TDCH-specific Hadoop attributes supplied in the TPT script. Once TDCH has validated the attribute values and filled in defaults for any missing attributes, TDCH will submit the job to the MapReduce framework. Once the map tasks have been initialized on the nodes in the Hadoop cluster, they will connect to the

DataConnector operator and begin transferring data. The HadoopProperties attribute can be used to specify one or more Hadoop properties and their values (separated by spaces) which will then be used by TPT when submitting the Hadoop command internally.

- The TDCH-TPT interface depends on the TDCH jar file. The latest certified TDCH jar file is included with the TPT installation and will be the one used by default. However, if the use of a different TDCH jar file is desired, the user can set the TDCH\_JARFILE environment variable to be the fully qualified filename of the desired TDCH jar. This one will then be used instead of the included TDCH jar file.

## Limitations to the TDCH-TPT Interface

- To utilize the TDCH-TPT interface, the node on which TPT is running must have the Hadoop client jars installed, as the DataConnector operator must be able to launch a MapReduce job via a call to the Hadoop CLI.
- The TDCH-TPT interface is only supported on the Linux platform.
- Because the DataConnector operator relies on TDCH to read from and write to Hadoop files and tables, many of the traditional DataConnector operator attributes are not supported alongside the TDCH-TPT interface. For example, when using the DataConnector producer, the FileName attribute is superseded by the TDCH-specific HadoopSourcePaths attribute. Similarly, the Format attribute is superseded by the TDCH-specific HadoopFileFormat attribute. If an unsupported attribute is submitted alongside TDCH-specific Hadoop attributes, the TPT job will fail.
- Due to TDCH's batch processing nature, many of the DataConnector operator's active data warehousing features are not supported when using the TDCH-TPT interface. For example, because the MapReduce job processes data out-of-order, the DataConnector's checkpoint/restart feature is unavailable when utilizing the TDCH-TPT interface. Similarly, because the TDCH job requires a single file or table name as an argument, the DataConnector operator's directory scan feature is unavailable when utilizing TDCH-TPT interface. If an unsupported feature is utilized alongside the TDCH-TPT interface, the TPT job will fail.
- When using the TDCH-TPT interface to process Hadoop files and tables, multiple instances of the DataConnector operator are not supported. If multiple instances of the DataConnector are defined, the TPT job will fail.

## Troubleshooting TDCH-TPT Jobs

In the scenario that a failure occurs during TDCH job setup or data transfer between TDCH and TPT, the TDCH log is available in the directory where TPT logs are generated. By default, this is `<installation directory>/logs`. The directory can also be changed by specifying the desired location on the `tbuild` command line with the `-L` option, or editing the `twbcfg.ini` file. See the *Teradata Parallel Transporter User Guide* (B035-2445) for additional information on redirecting log files. The name of the TDCH log will be `TDCH-TPT_log_<job_id>.txt`, where `<job_id>` is the job's process ID. For more information, see the MapReduce logs via the JobTracker's web interface or by navigating to the Hadoop installation's `userlogs` directory.

## Access Module Instances

There are three ways access modules are engaged when multiple FILE\_READER instances are used:

- When the FileName attribute specifies a single file name, only the access module for the main instance will be sent Open and Read requests. The worker instances will be sent other requests (such as Init and Shutdown), but not Open and Read requests.
- When the FileName attribute includes a wildcard, one or more access modules will be sent Open and Read requests, depending upon how many files match the wildcard syntax. For example, if there are 5 FILE\_READER instances but only 3 files match the wildcard syntax, then 3 access modules will be sent Open and Read requests. Access modules for the other two instances will be sent other requests (such as Init and Shutdown), but not Open and Read requests.
- When the FileName attribute is not defined but the AccessModuleInitStr attribute is defined, then the access modules for all FILE\_READER instances will be sent Open and Read requests. It is up to the access module itself to know which file to open (possibly based on information in the initialization string).

## Enhanced LOB Support

As a consumer and producer, the DataConnector operator supports enhanced LOB functionality.

For more information, see [Enhanced LOB Support](#).

## Operational Considerations

### File Size Restrictions

Windows, Oracle Solaris running on a SPARC system, AIX, z/OS, and Linux operating systems place no 2-gigabyte file size restriction, provided system parameters are appropriately set.

### z/OS Considerations

- The vigil window feature is not available for z/OS PDS files.
- Load distribution of input files across instances are based on file count rather than file size.

### VSAM Files on z/OS

If a VSAM file is changed or updated between checkpoints, the results of the restart are unreliable.

## VSAM Alternate Indexes

KSDS/ESDS alternate indexes can be read sequentially using the pathname. But checkpoint/restart is not supported for KSDS/ESDS alternate indexes.

## End of Record Markers for Text Formatted Data

There are 2 forms of the end-of-record (EOR) marker.

- On UNIX, it is a single byte LF (x'0A').
- On Windows, it is a two byte CRLF (x'0D0A').

The DataConnector operator examines the input file and uses the EOR that it finds. Files may be FTP'd between platforms.

## Considerations when Processing Hadoop Files and Tables

- When using the HDFS API Interface, the HadoopHost value must point to the NameNode of the Hadoop cluster.
- When using the TDCH-TPT Interface, the HadoopHost value must be the host name or IP address of the node on which TPT is running. This host name or IP address must be reachable from all the DataNodes in the Hadoop cluster.
- Both the HDFS API Interface and the TDCH-TPT Interface require that the node on which TPT is running has the Hadoop client jars installed and the HDFS API also requires a copy of the Hadoop Cluster Configuration files.
- When using the HDFS API Interface, the version of the Hadoop client jars must match the version of the Hadoop jars on the NameNode of the Hadoop cluster defined by the HadoopHost attribute.
- The HDFS API requires that the following environment variables must be set:
  - JAVA\_HOME = Root Location of the Java JDK

**Note:**

- The HDFS API uses JAVA JNI.
  - The 64-bit version of Java.
- 
- HADOOP\_HOME = Root Location of the Hadoop Client Jar Files and Hadoop Configuration files.
- The HDFS API can use the following optional environment variables:
    - LIBHDFS\_OPTS = extra options to pass into the JVM.LIBHDFS\_JVM\_PATH = full-path of jvm.dll (vm.so).
    - If LIBHDFS\_JVM\_PATH is not provided, then JAVA\_HOME is used to locate the jvm.dll.
    - LIBHDFS\_CLASSPATH = classpath for jni jvm.

- LIBHDFS\_CLASSPATH takes precedence over CLASSPATH.
  - If neither CLASSPATH or LIBHDFS\_CLASSPATH is provided, HADOOP\_HOME or HADOOP\_PREFIX will be used to create the CLASSPATH.
  - If CLASSPATH or LIBHDFS\_CLASSPATH contains syntax that is incompatible with JNI (that is, “\*”), they will be discarded and HADOOP\_HOME or HADOOP\_PREFIX will be used to create the classpath.
  - CLASSPATH = class path for jni jvm.
- 

**Note:**

Not wise to use because it is usually not set correctly for JNI.

---

- JAVA\_LIBRARY\_PATH = Path to hadoop.dll for java native method support.
- When using the TDCH-TPT Interface, the following environment variables must be defined. Before launching the TDCH job, the TDCH-TPT interface checks to see if the environment variables are set, and if not the TDCH-TPT interface exports the environment variables with the default values defined in the following list. See the [Teradata Connector for Hadoop tutorial](#) for more information about the environment required by TDCH.
  - HADOOP\_HOME – location of the Hadoop libraries; this value defaults to '/usr/lib/hadoop'.
  - HIVE\_HOME – location of the Hive libraries; this value defaults to '/usr/lib/hive'.
  - HCAT\_HOME – location of the HCatalog libraries; this value defaults to '/usr/lib/hcatalog'.
  - HIVE\_LIB\_JARS – comma separated list of jar files required by TDCH; if not set the TDCH-TPT interface will build this list using the jars found in the \$HCAT\_HOME/lib and \$HIVE\_HOME/lib.
  - HADOOP\_CLASSPATH – semicolon separated list of jar files and directories required by TDCH; if not set the TDCH-TPT interface will build this list using the jars found in \$HCAT\_HOME/lib and \$HIVE\_HOME/lib, as well as the directory \$HIVE\_HOME/conf.

## DataConnector Operator Events

The following table lists the event codes and describes the data that the DataConnector operator passes to the notify exit routine for each event. The information in this table is also sent to the system log.

---

**Note:**

To support future enhancements, always verify that your *notify* exit routines ignore invalid or undefined event codes, and that they do not cause the operator to terminate abnormally.

---

Event Code	Event	Notification Level			Event Description
		Low	Med	High	
0	Initialize	Yes			Signifies successful processing of the notify feature. • Operator handle – 4-byte unsigned integer

Event Code	Event	Notification Level			Event Description
		Low	Med	High	
					<ul style="list-style-type: none"> <li>• Operator number – 4-byte unsigned integer</li> <li>• Operator count – 4-byte unsigned integer</li> <li>• Utility ID – 4 byte unsigned integer</li> <li>• Task ID – 32-character (maximum) array</li> <li>• Time stamp – 26-character (fixed size) array with the format YYYY-MM-DDHH:MM:SS.SSSSSS</li> </ul> <p>The following offset fields contain the byte offsets to data items in the event record that are null terminated strings. Given the address of the event record, the address of such a string can be obtained from the offset.</p> <ul style="list-style-type: none"> <li>• Job ID offset – 4-byte unsigned integer</li> <li>• Version ID offset – 4-byte unsigned integer</li> <li>• Utility name offset – 4-byte unsigned integer</li> <li>• User string offset – 4-byte unsigned integer</li> </ul>
1	Directory Scan Complete			Yes	<p>Signifies successful processing of a scan for file names in a directory.</p> <ul style="list-style-type: none"> <li>• Operator number – 4-byte unsigned integer</li> <li>• Task name – 32-character (maximum) array</li> <li>• File count – 4-byte unsigned integer</li> <li>• Time stamp – 26-character (fixed size) array with the format YYYY-MM-DDHH:MM:SS.SSSSSS</li> </ul> <p>The following offset fields contain the byte offsets to data items in the event record that are null terminated strings. Given the address of the event record, the string address of each data item can be obtained from its offset.</p> <ul style="list-style-type: none"> <li>• Script name offset – 4-byte unsigned integer</li> <li>• Job name offset – 4-byte unsigned integer</li> <li>• Job ID offset – 4-byte unsigned integer</li> <li>• File list offset – 4-byte unsigned integer (Note: there are "File Count" NULL terminated file name strings arranged end-to-end in the file list)</li> </ul>
2	File Read Complete			Yes	<p>Signifies successful processing of a file in the file list.</p> <ul style="list-style-type: none"> <li>• Operator number – 4-byte unsigned integer</li> <li>• Task ID – 32-character (maximum) array</li> <li>• Record count – 4-byte unsigned integer</li> <li>• File size – 4-byte unsigned integer</li> </ul> <p>The following offset fields contain the byte offsets to data items in the event record that are null terminated strings. Given the address of the event record, the address of such a string can be obtained from the offset.</p> <ul style="list-style-type: none"> <li>• Script name offset – 4-byte unsigned integer</li> <li>• Job name offset – 4-byte unsigned integer</li> <li>• Job ID offset – 4-byte unsigned integer</li> <li>• File name offset – 4-byte unsigned integer</li> </ul>

Event Code	Event	Notification Level			Event Description
		Low	Med	High	
3	Checkpoint			Yes	<p>Signifies successful processing of a checkpoint.</p> <ul style="list-style-type: none"> <li>• Operator number – 4-byte unsigned integer</li> <li>• Task ID – 32-character (maximum) array</li> </ul> <p>The following offset fields contain the byte offsets to data items in the event record that are null terminated strings. Given the address of the event record, the string address of each data item can be obtained from its offset.</p> <ul style="list-style-type: none"> <li>• Script name offset – 4-byte unsigned integer</li> <li>• Job name offset – 4-byte unsigned integer</li> <li>• Job ID offset – 4-byte unsigned integer</li> </ul>
4	Terminate	Yes			<p>Signifies successful termination of the notify feature.</p> <ul style="list-style-type: none"> <li>• Operator number – 4-byte unsigned integer</li> <li>• Task ID – 32-character (maximum) array</li> <li>• Return code – 4-byte unsigned integer</li> <li>• Time stamp – 26-character (fixed size) array with the format YYYY-MM-DDbHH:MM:SS.SSSSSS</li> </ul>

# DDL Operator

## DDL Operator Capabilities

The main task of DDL operator is to send SQL statements to the database to perform setup operations prior to executing load or export tasks.

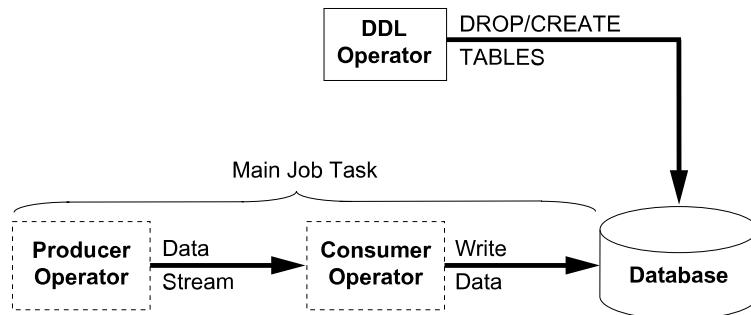
The DDL operator is frequently used to create or drop error tables or indexes required by Teradata PT operators that will perform the main job tasks. You can, for example submit an INSERT...SELECT statement using the DDL operator to move data from one table to another. The DDL operator is also used to perform mini-batch loading tasks.

The DDL operator is neither a producer nor consumer operator. It does not retrieve data records from the Teradata PT data stream nor put records into the data stream. An APPLY statement is used in a Teradata PT job script in which a DDL operator is defined.

### Note:

Because conditions in the database may change between uses of a particular job script, you should always evaluate the need to add or change the DDL operator in the script before running a particular Teradata PT job.

The following figure shows the DDL operator interface.



## The DDL Operator Function in a Teradata PT Job

When you use the DDL operator in a Teradata PT job, Teradata PT directs an instance of the DDL operator to:

- Log on to the database using your user name, password, database name, and account ID information specified in the job script.
- The operator terminates when scripts contain invalid or unsupported statements.

- Submit a request (this can be one statement or multiple statements) to the database and processes returned parcels from these statements.
- Issue, if an error occurs, an error message and terminate, unless the error is specified in the ErrorList attribute, in which case the DDL operator ignores the error and continues processing the job.
- Log off the database.
- Put out an end status indicating how many requests succeeded, failed, or were ignored.
- Resume, in the case of a restart, at the group that was being executed, but failed to complete, in the previous execution.

## Syntax

Use the following specifications and attributes to define the DDL operator in a Teradata PT job script.

## Required Specifications

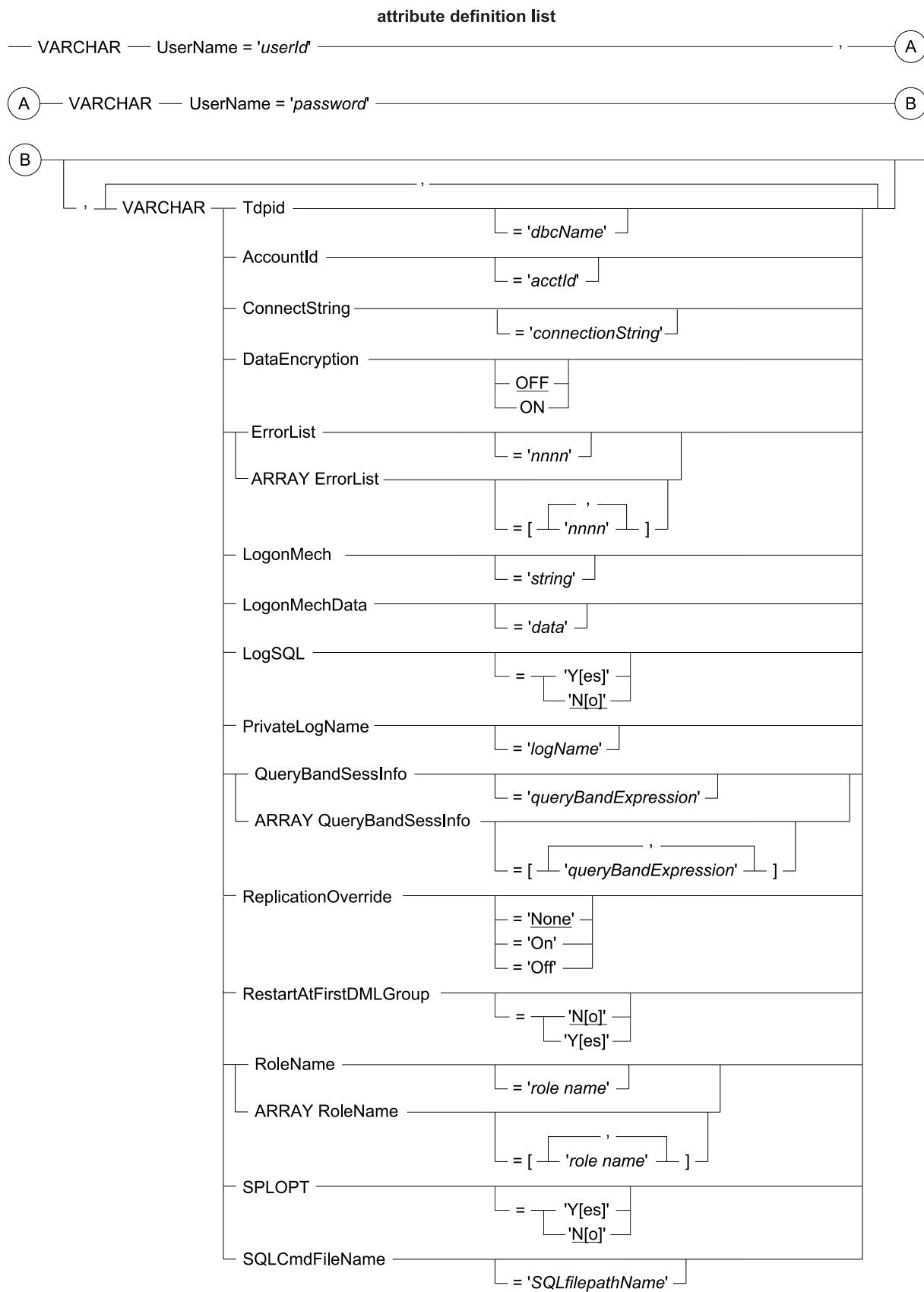
The following specifications are required to define a job. See [Required and Optional Attributes](#) for information about defining required and optional operator attributes.

**Required Syntax for the DDL Operator**

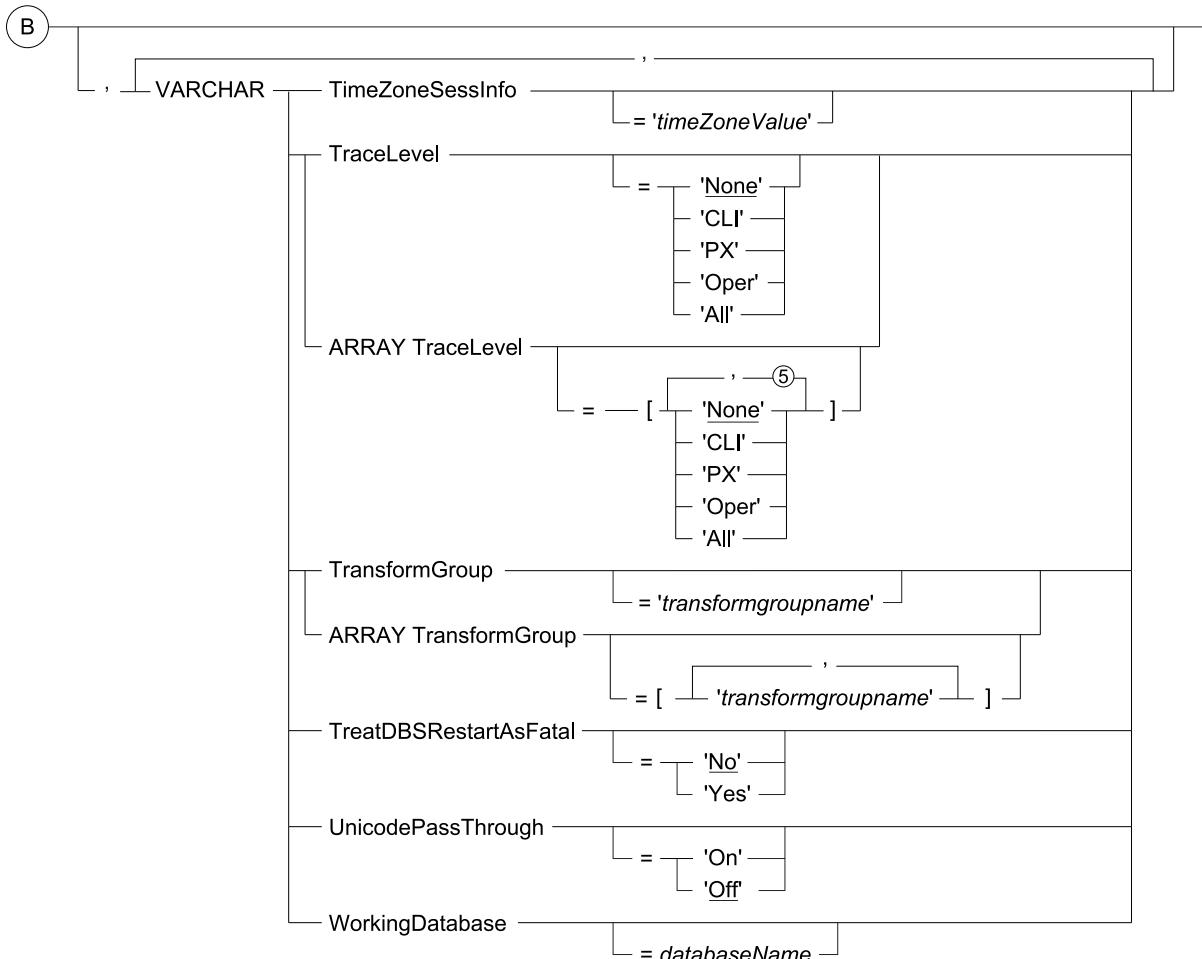
Specification	Description
TYPE	Operator type. Always DDL for the DDL operator.
UserName	Operator attribute that provides the name of the user for the DDL operator logon sessions.
UserPassword	Operator attribute that provides the password associated with the user name.

## Required and Optional Attributes

Use the attribute definition list syntax in the Teradata PT DEFINE OPERATOR statement to declare the required and optional attribute values for the DDL operator. See the DDL Operator Attribute Definitions table in this section for descriptions of the attributes in the following syntax diagrams. A generic example of the operator follows the attribute descriptions.



## attribute definition list (continued)



where:

#### DDL Operator Attribute Definitions

Syntax Element	Description
AccountId = <i>acctId</i>	Optional attribute that specifies the account associated with the specified user name. If omitted, it defaults to the account identifier of the immediate owner database.
ARRAY	Optional keyword that specifies more than one attribute value.
ConnectionString = ' <i>connectionString</i> '	Optional attribute that specifies the connection string. The connection string will be passed to CLI. CLI will validate the connection string. For information on connection string, see <i>Teradata® Call-Level Interface Version 2 Reference for Workstation-Attached Systems</i> , B035-2418.

Syntax Element	Description
	<p><b>Note:</b> The TPT Connection String feature is available on all platforms, except on z/OS.</p>
DataEncryption = ' <i>option</i> '	<p>Optional attribute that enables full encryption of SQL requests, responses, and data. Valid values are:</p> <ul style="list-style-type: none"> <li>• ON = All SQL requests, responses, and data are encrypted</li> <li>• OFF = No encryption occurs (default)</li> </ul>
ErrorList = ' <i>nnnn</i> '	<p>Optional attribute that specifies a list of database errors (by number) to ignore. Using this attribute suppresses known database errors, preventing the termination of a job. This attribute overrides the default operator behavior that terminates a job whenever an error returns from the Teradata server when a DDL statement is issued. The following example prevents the DDL operator from returning a bad status code when database error 3807 occurs during a table drop:</p> <pre>DEFINE OPERATOR DDLOP DESCRIPTION 'DDL OPERATOR' . ATTRIBUTES (     VARCHAR ErrorList = '3807', ; </pre> <p>You can also use the ErrorList attribute to suppress multiple error codes.</p>
LogonMech = ' <i>string</i> '	<p>Optional attribute that specifies which logon mechanism to use.</p> <p><b>Note:</b> Specification of this attribute may be required for some authentication methods. The job terminates if the attribute exceeds 8 bytes. For information on specification requirements for LogonMech "Logon Security" in <i>Teradata Parallel Transporter User Guide</i> (B035-2445).</p>
LogonMechData = ' <i>data</i> '	<p>Optional attribute that passes along additional logon data.</p> <p><b>Note:</b> Specification of this attribute is required for some external authentication methods. For information on specification requirements for LogonMechData "Logon Security" in <i>Teradata Parallel Transporter User Guide</i> (B035-2445).</p>
LogSQL = ' <i>option</i> '	<p>Optional attribute that controls how much of the job's SQL to enter into the log. Valid options are:</p> <ul style="list-style-type: none"> <li>• 'Yes' = output the full SQL to the log. The maximum length is 1M.</li> </ul>

Syntax Element	Description
	<ul style="list-style-type: none"> <li>'No' = do not output SQL to the log.</li> <li>No value or attribute omitted = accept the pre-defined limit, which displays up to 32K of SQL if all the SQL is less than 32K. If the SQL to be logged exceeds 32K, truncate the display to 32K.</li> </ul>
PrivateLogName = ' <i>logName</i> '	<p>Optional attribute that specifies the name of a log that is maintained by the Teradata PT Logger inside the public log. The private log contains all the output provided by the operator.</p> <p>The private log can be viewed using the tlogview command as follows, where <i>jobId</i> is the Teradata PT job name and <i>privateLogName</i> is the value for the operator's PrivateLogName attribute:</p> <pre>tlogview -j jobId -f privateLogName</pre> <p>If the private log is not specified, all output is stored in the public log. For more information about the tlogview command, see <a href="#">Teradata PT Utility Commands</a>.</p>
QueryBandSessInfo = ' <i>queryBandExpression</i> '	<p>Optional attribute that specifies the Query Band for the duration of the job sessions.</p> <p>The <i>queryBandExpression</i> is a set of name=value pairs, separated by a semicolon and ending with a semicolon. The user defines the Query Band expression, which will look similar to the following example:</p> <pre>'org=Finance;load=daily;location=west;'</pre> <p>QueryBandSessInfo may also be specified as an ARRAY attribute. For information on the rules for creating a Query Band expression, see <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i>, B035-1144 and <i>Teradata Vantage™ - SQL Data Definition Language Detailed Topics</i>, B035-1184.</p> <p>The value of the QueryBandSessInfo attribute is displayed in the DDL operator private log.</p> <p>Use of the QueryBandSessInfo attribute is subject to the following rules:</p> <ul style="list-style-type: none"> <li>By default, Query Band is off until a valid value appears for the QueryBandSessInfo attribute.</li> <li>If the QueryBandSessInfo attribute contains a value, the DDL operator constructs the necessary SET QUERY BAND SQL and issues it as part of the DDL operator SQL sessions to communicate the request to the database.</li> <li>The DDL operator does not check the Query Band expression, but passes the expression to the database as is.</li> <li>If the version of database against which the job is being run does not support the Query Band feature, no Query Banding will take place. However, the operator will ignore the error and run the rest of the job.</li> <li>If there is a syntax error in the Query Band expression, the database returns an error. The DDL operator then terminates the job and report the error to the user.</li> </ul>
ReplicationOverride = ' <i>option</i> '	Optional attribute that overrides the normal replication services controls for an active session.

Syntax Element	Description
	<p>Valid values:</p> <ul style="list-style-type: none"> <li>‘On’ = Override normal replication services controls for the active session.</li> <li>‘Off’ = Override of normal replication services is turned off for the active session (when change data capture is active).</li> <li>‘None’ = (Default) No override request is sent to the database.</li> </ul> <p>For more information, see <i>Teradata Replication Services Using Oracle GoldenGate</i>.</p> <p><b>Note:</b></p> <p>The user ID that is logged in by the DDL operator must have the REPLCONTROL privilege when setting the value for this attribute.</p>
RestartAtFirstDMLGroup = 'option'	<p>Optional attribute that specifies which DML statement the DDL operator will resume at after a restart.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>'Yes' = restart at the first DML statement in the group</li> <li>'No' = restart at the DML statement that failed in the previous run (default)</li> </ul> <p>The following example illustrates the behavior:</p> <pre>CREATE TABLE A (...); CREATE TABLE B (...); CREATE TABLE C (...);</pre> <p>If the job fails during creation of table C, for example, the default behavior when restarting the job is that the DDL operator will resume at the 'CREATE TABLE C' DML statement. However, if RestartAtFirstDMLGroup is set to 'Yes', the DDL operator will resume at the 'CREATE TABLE A' DML statement.</p>
RoleName = 'role name'	<p>Optional attribute that implements security in a database environment. The operator will prepend the value with "SET ROLE ". The syntax will be sent to the database as follows:</p> <pre>SET ROLE &lt;role name&gt;;</pre> <p>For example:</p> <pre>SET ROLE A1;</pre> <p>For details of SET ROLE command use, see <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i>, B035-1144.</p> <p>The VARCHAR ARRAY can specify more than one value, for example:</p> <pre>VARCHAR ARRAY RoleName = [ 'role name1', 'role name2' ],</pre> <p>The operator will send the request to the database on the SQL session after the session is connected.</p>

Syntax Element	Description
	<p><b>Note:</b></p> <ul style="list-style-type: none"> <li>The operator does not validate the value for this attribute. The operator passes the value to the database as is. The database will validate the value. The operator will terminate the job with an error when the validation fails.</li> <li>C-style comments are allowed in the value and will be passed to the database.</li> <li>ANSI-style comments are not supported in the value. The operator can terminate the job with a syntax error when the value contains an ANSI-style comment.</li> <li>A semicolon is not allowed in the value, because the operator allows only a single statement per request. The operator will terminate the job with an operator error when the value contains a semicolon.</li> </ul>
SPLOPT = ' <i>option</i> '	<p>Optional attribute that supports creating, replacing, and dropping an inline stored procedure using the DDL operator. The DDL operator does not execute a stored procedure. The stored procedure can be fetched with the SHOW PROCEDURE statement.</p> <p>For stored procedure commands that exceed 64K, Teradata PT sends the request over in blocks of 64K, one at a time.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>'Yes' = an internal flag is set telling the database to store the TDSP (default).</li> <li>'No' = no internal flag is set telling the database to store the TDSP.</li> </ul>
SQLCmdFileName = <i>SQLfilepathname</i>	<p>Optional VARCHAR attribute that specifies the complete SQL file path name. In the provided &lt;SQLFile&gt;, each SQL command must end with a semicolon.</p>
Tdpld = ' <i>dbName</i> '	<p>Optional attribute that specifies the name of the database machine (non-mainframe platforms) or TDP (mainframe platforms) for the DDL operator job.</p> <p>The <i>dbName</i> can be up to 256 characters and can be a domain server name.</p> <p>If you do not specify the value for the Tdpld attribute, the operator uses the default TdpID established for the user by the system administrator.</p>
TimeZoneSessInfo = ' <i>timeZoneValue</i> '	<p>Optional attribute that allows you to change the default time zone displacement for the duration of the operator's job session.</p> <p>When you provide a value for this attribute, the operator will build the SET TIME ZONE &lt;<i>timeZoneValue</i>&gt;; SQL request.</p> <p>The operator will send the request to the database on the SQL session after the session is connected.</p> <p>Here are some examples:</p> <ul style="list-style-type: none"> <li><b>Example 1:</b> This example sets the session default time zone displacement to LOCAL, which is the system default time zone:  <code>VARCHAR TimeZoneSessInfo = 'LOCAL'</code></li> <li><b>Example 2:</b> This example sets the session default time zone displacement to USER, which is the default time zone for the logged on user:  <code>VARCHAR TimeZoneSessInfo = 'USER'</code></li> </ul>

Syntax Element	Description
	<ul style="list-style-type: none"> <li><b>Example 3:</b> This example sets the session default time zone displacement to a simple constant time zone string expression:  <pre>VARCHAR TimeZoneSessInfo = '''America Pacific'''</pre> </li> </ul> <p><b>Note:</b>  Any single quote character ('') inside the value must be entered as two consecutive single quote characters in a TPT job script. This ensures the correct value will be sent to the database.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>The operator does not validate the value for this attribute. The operator passes the value to the database as is. The database will validate the value. The operator will terminate the job with an error when the validation fails.</li> <li>C-style comments are allowed in the value and will be passed to the database.</li> <li>ANSI-style comments are not supported in the value. The operator will terminate the job with a syntax error when the value contains an ANSI-style comment.</li> <li>A semicolon is not allowed in the value, because the operator allows only a single statement in the "SET TIME ZONE SQL" request. The operator will terminate the job with an operator error when the value contains a semicolon.</li> </ul> <p>For more information on SET TIME ZONE SQL, see <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i>, B035-1144.</p>
TraceLevel = 'level'	<p>Optional attribute that specifies the types of diagnostic messages that are written by each instance of the operator to the public log (or private log, if one is specified using the PrivateLogName attribute). The diagnostic trace function provides more detailed information in the log file to aid in problem tracking and diagnosis.</p> <p>The trace levels are:</p> <ul style="list-style-type: none"> <li>'None' = (Default) TraceLevel turned off.</li> <li>'CLI' = enables the tracing function for CLI-related activities (interaction with the database)</li> <li>'PX' = enables the tracing function for activities related to the Teradata PT infrastructure</li> <li>'Oper' = enables the tracing function for operator-specific activities</li> <li>'All' = enables tracing for all the mentioned activities</li> </ul> <p>The VARCHAR ARRAY can specify more than one value, for example:</p> <pre>VARCHAR TraceLevel = 'CLI' VARCHAR TraceLevel = 'OPER' VARCHAR ARRAY TraceLevel = [ 'CLI' ] VARCHAR ARRAY TraceLevel = [ 'CLI', 'OPER' ]</pre>

Syntax Element	Description
	<p><b>Note:</b></p> <p>The TraceLevel attribute is provided as a diagnostic aid only. The amount and type of additional information provided by this attribute changes to meet evolving needs from release to release.</p>
TransformGroup = 'transformgroupname'	<p>Optional attribute that supports changing the active transform for Teradata Complex Data Types (CDTs). The value is the <i>&lt;udt name&gt; &lt;transform group name&gt;</i>, and the operator will prepend the hard-coded string "SET TRANSFORM GROUP FOR TYPE ". The syntax sent to RDBMS is as follows:</p> <pre>SET TRANSFORM GROUP FOR TYPE &lt;udt name&gt; &lt;transform group name&gt;;</pre> <p>For example:</p> <pre>"SET TRANSFORM GROUP FOR TYPE JSON CHARACTER SET LATIN TD_JSON_VARCHAR;"</pre> <p>The VARCHAR ARRAY can specify more than one value, for example:</p> <pre>VARCHAR ARRAY TransformGroup = [ 'JSON CHARACTER SET LATIN TD_JSON_VARCHAR',                                 'ST_Geometry TD_Geo_VARCHAR' ],</pre> <p>The operator will send the request to the database on the SQL session after the session is connected.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>The operator does not validate the value for this attribute. The operator passes the value to the database as is. The database will validate the value. The operator will terminate the job with an error when the validation fails.</li> <li>C-style comments are allowed in the value and will be passed to the database.</li> <li>ANSI-style comments are not supported in the value. The operator can terminate the job with a syntax error when the value contains an ANSI-style comment.</li> <li>A semicolon is not allowed in the value, because the operator allows only a single statement per request. The operator will terminate the job with an operator error when the value contains a semicolon.</li> </ul>
TreatDBSRestartAsFatal= 'option'	<p>Optional attribute that tells the operator whether to terminate the job when a database restart occurs.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>'No' ('N') = The operator will not terminate if a database restart occurs (default). The database restart will be treated as a retryable one.</li> <li>'Yes' ('Y') = The operator will terminate if a database restart occurs.</li> </ul>
UnicodePassThrough = 'value'	<p>Optional attribute that tells the operator to enable or disable the Unicode Pass Through feature.</p> <p>Valid values:</p> <ul style="list-style-type: none"> <li>'On' = Enable the Unicode Pass Through feature in the operator.</li> <li>'Off' = (Default) Disable the Unicode Pass Through feature in the operator.</li> </ul>

Syntax Element	Description
	<p><b>Note:</b> When a TPT job is using the UTF8 or UTF16 session character set, the UnicodePassThrough attribute can be set to 'On' to allow the operator to send data with Unicode pass through characters to the database.</p>
Username = 'userId'	<p>Attribute that specifies the database user name.</p> <p><b>Note:</b> Use of this attribute is not compatible with some external authentication logon methods. For more information on UserName specification requirements, see "Logon Security" in <i>Teradata® Parallel Transporter User Guide</i>, B035-2445.</p>
UserPassword = 'password'	<p>Attribute that specifies the password associated with the user name.</p> <p><b>Note:</b> Use of this attribute is not compatible with some external authentication logon methods. For more information on password specification requirements, see "Logon Security" in <i>Teradata® Parallel Transporter User Guide</i>, B035-2445.</p>
VARCHAR	Keyword that specifies VARCHAR as the data type of the defined attribute.
WorkingDatabase = 'databaseName'	<p>Optional attribute that specifies a database other than the logon database as the default database.</p> <p>The name of the database that is specified with this attribute is used in the SQL DATABASE statement that is sent by the operator immediately after connecting the two SQL sessions.</p> <p>If WorkingDatabase is not specified, the default database associated with the logged on user is assumed for all unqualified table names.</p>

## Usage Notes

### Supported SQL Statements

The DDL operator supports every SQL statement except statements that:

- Return data to the operator, such as the SELECT, HELP, and SHOW statements.

The CREATE REPLICATION GROUP statement is supported. The user needs the group id and/or the token after a replication group is created.

The INSERT...SELECT statement is supported. It returns the total number of rows being inserted in the table.

- Require the operator to send data back to the database.

Because standalone operators do not send data to or retrieve data from data streams, the DDL operator does not support the USING clause with the INSERT, UPDATE, and DELETE DML SQL statements.

**Note:**

The Stream operator and the Update operator automatically employ the USING clause where needed.

All data values used by the SQL statement used by the DDL operator must be hard-coded into the submitted SQL statements.

## Specifying DDL Statements in the Teradata PT APPLY Statement

The SQL statements that the DDL operator supports are specified in the Teradata PT APPLY statement.

The following examples show how to specify DDL statements in the APPLY statement:

- One SQL statement per group

```
APPLY
  'ddl stmt1',
  'ddl stmt2',
  .....
  'ddl stmt3'
TO OPERATOR (operator specification)
```

- Multiple SQL statements in a group, but only one group

```
APPLY
  ('ddl stmt1','ddl stmt2', ... , 'ddl stmtn')
TO OPERATOR (operator specification)
```

- Multiple SQL statements per group, and multiple groups

```
APPLY
  ('ddl stmt11','ddl stmt12', ... , 'ddl stmt1n'),
  .
  .
  .
  ('ddl stmtx1','ddl stmtx2', ... , 'ddl stmtxm')
TO OPERATOR (operator specification)
```

If more than one statement is specified in one group, then the DDL operator combines all statements into one single multistatement request and sends it to the database as one implicit transaction. This means that any statement failure or any error rolls back the entire transaction.

The SQL statements are executed by groups in the order they are specified in the APPLY statement. Currently, the maximum number of SQL statement groups that can be specified in the APPLY statement

is 1024. If the number of SQL statement groups is greater than 1024, Teradata PT terminates the job with the TPT05537 error message.

## Grouping SQL Statements in the Teradata PT APPLY Statement

You can group several SQL statements together to perform a desired logical database task and still take advantage of the automatic rollback feature in Vantage in case of any statement failures or any errors occurring during the transaction.

However, you should have one SQL statement per group if you desire to execute each statement in its own transaction.

When multiple SQL statements are specified in one DML group in the APPLY statement, the database enforces the rule that if the group contains a DDL statement, it must be the last statement in the implicit transaction, which means the last statement in a group.

Therefore, given that the information in parentheses represents a group, the validity of the statements can be determined as follows:

- Group 1: (DDL) is valid.
- Group 2: (DDL, DDL) is invalid because only one DDL statement is allowed.
- Group 3: (DML, DML, DDL) is valid.
- Group 4: (DML, DML, DML) is valid, even though the group contains no DDL statement.
- Group 5: (DML, DDL, DML) is invalid because the DDL statement is not the last statement in the group.

If a script contains unsupported or invalid statements, the job terminates so the script can be fixed before continuing.

## Specifying SQL Statements Using the SQLCmdFileName Attribute

There is no limitation on the number of SQL statements that can be specified in the file given in the SQLCmdFileName attribute.

Every SQL statement should be terminated with a semicolon at the end.

SQL comments are not supported in the file. C style comments are supported but will be stripped off before the request is sent to the database.

The statements supported in the file are regular SQL statements and substitutions using '@' on job variables is not supported.

## Combination of Both APPLY and SQLCmdFileName

You can specify SQL statements either in the APPLY Statement and/or in a file specified in the SQLCmdFileName attribute of the DDL operator. The order of execution will be:

1. SQL statements specified in the <SQLFile>.
2. SQL statements specified in the TPT script.

If the file given in the SQLCmdFileName attribute is empty and no SQL statement was provided in the APPLY statement, the job would result in an error.

## Restrictions and Limitations

This section describes the restrictions and limitations when using the DDL operator.

- The DDL operator is neither a producer nor a consumer operator.  
The DDL operator is a standalone operator. The DDL operator cannot retrieve data records from the Teradata PT data stream and send them to the database. The DDL operator cannot export rows from the database.
- The DDL operator uses one session; it does not support multiple sessions.
- The DDL operator uses one instance; it does not support multiple instances.
- By default, if the DDL operator encounters an error, the DDL operator treats the error as a fatal error and terminates the job. You can override this behavior by using the DDL operator's ErrorList attribute.
- The DDL operator supports every SQL statement, except statements that return data to the operator, such as the SELECT, HELP, and SHOW statements. There are two exceptions:
  - The CREATE REPLICATION GROUP statement is supported. The user needs the group id and/or the token after a replication group is created.
  - The INSERT...SELECT statement is supported; it returns the total number of rows being inserted in the table.
- A USING clause cannot be specified with any SQL statements that send data to the database.
- All data values used by the SQL statements must be hard-coded into the SQL statements.
- The DDL operator restarts at the beginning of the group of SQL statements whose execution is interrupted by an abnormal termination.
- The DDL operator can send SQL statements up to 1MB to the database.
- The DDL operator does not require a database load slot.

## Operational Considerations

### Checkpointing and Restarting

Because SQL statements are sent to the database by group in the order in which they are specified in the Teradata PT APPLY statement, the DDL operator can take a checkpoint after each group is executed. A group can contain one or more SQL statements. A checkpoint, with respect to the DDL operator, marks the last group of DDL/DML SQL statements to execute successfully.

The DDL operator restarts at the beginning of the group of SQL statements whose execution is interrupted by an abnormal termination. If the interrupted group has only one SQL statement, the DDL operator restarts at that statement.

If the last request was successful prior to a restart, the operator can resume at the next request in line. If the last request failed prior to a restart, then the operator resumes at the failed request.

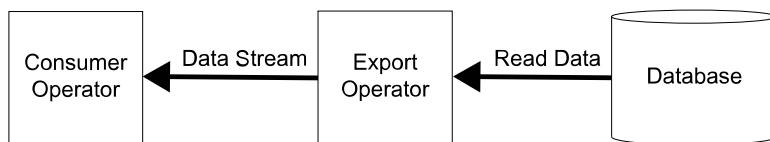
# Export Operator

## Export Operator Capabilities

The Export operator, a producer operator, uses Teradata FastExport protocol to extract large amounts of data at high speeds from the database using block transfers over multiple sessions.

The Export operator reads data from one or more tables in the database.

The following figure shows the Export operator interface.



## The Export Operator Function in a Teradata PT Job

When you use the Export operator in a Teradata PT job, the Export operator does the following:

1. Logs on to the database, using your user name, password, database name, and account ID information specified in the job script.
2. Sends the SELECT request in the SelectStmt attribute information and retrieve the exported data from the database.
3. Logs off from the database.
4. If the Export operator job is successful, the job completes and returns information about the job, such as:
  - Total number of records exported from the database.
  - Records exported by each instance.
5. If the job is unsuccessful, the job terminates and provides information about the job so that you can correct any problem and restart the job.

## FastExport Utility and the Export Operator

The following table lists the operating features and capabilities of the Teradata FastExport utility and indicates whether such features and capabilities are supported by the Export operator.

FastExport Utility Feature	Teradata PT Operator Support
Access Modules	Supported, using the DataConnector operator

FastExport Utility Feature	Teradata PT Operator Support
ANSI Date	Supported
Blocksize Specification	Supported
Character sets	Supported, using the USING CHARACTER SET clause before the DEFINE JOB statement
Checkpoint/restart	Supported
Configuration file	Supported, using the tbuild command line job attribute option (-v)
DECIMALDIGITS option	Supported
DISPLAY command	Not supported
Environment variables	Not supported
IF-ELSE-ENDIF commands	Not supported
Indicator mode	Supported, using the DataConnector operator
INMOD routines	Not supported
IMPORT command	Not supported
Maximum/minimum sessions	Supported
Multiple SQL SELECT statements	Supported, with same layout, not supported with different layouts
Record formats	Supported, using the DataConnector operator
Nonindicator mode	Supported, using the DataConnector operator
Notify	Supported
OUTFILE option	Supported, using the DataConnector operator
OUTLIMIT specification	Supported
OUTMOD routines	Supported, using the FastExport OUTMOD Adapter operator
ROUTE MESSAGES command	Not supported
RUN FILE command	Supported by Teradata PT script language
SET command	Not supported
Show version information	Supported
No Spooling	Supported
SQL SELECT Statement	Supported

FastExport Utility Feature	Teradata PT Operator Support
SQL Statements (other such as CREATE TABLE, DROP TABLE, and so on)	Supported, using the DDL operator
SQL DATABASE Statement	Supported
SYSTEM Command	Supported, using the OS Command operator
Tenacity	Supported
User-defined variables	Limited support via script syntax

## Syntax

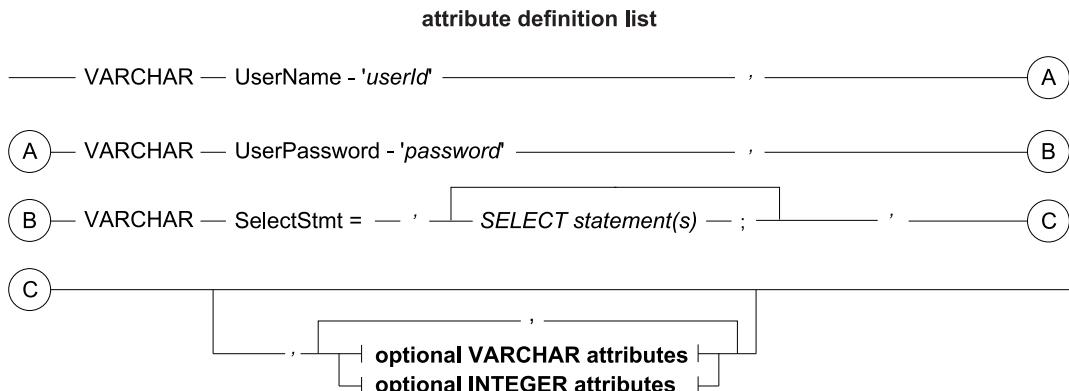
Use the following specifications and attributes to the Export operator in a Teradata PT job script.

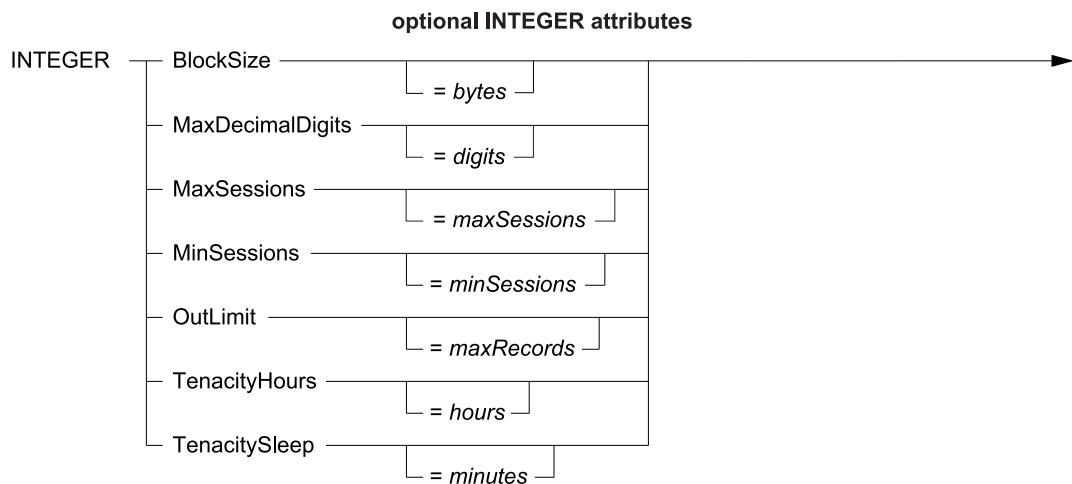
### Required Syntax for the Export Operator

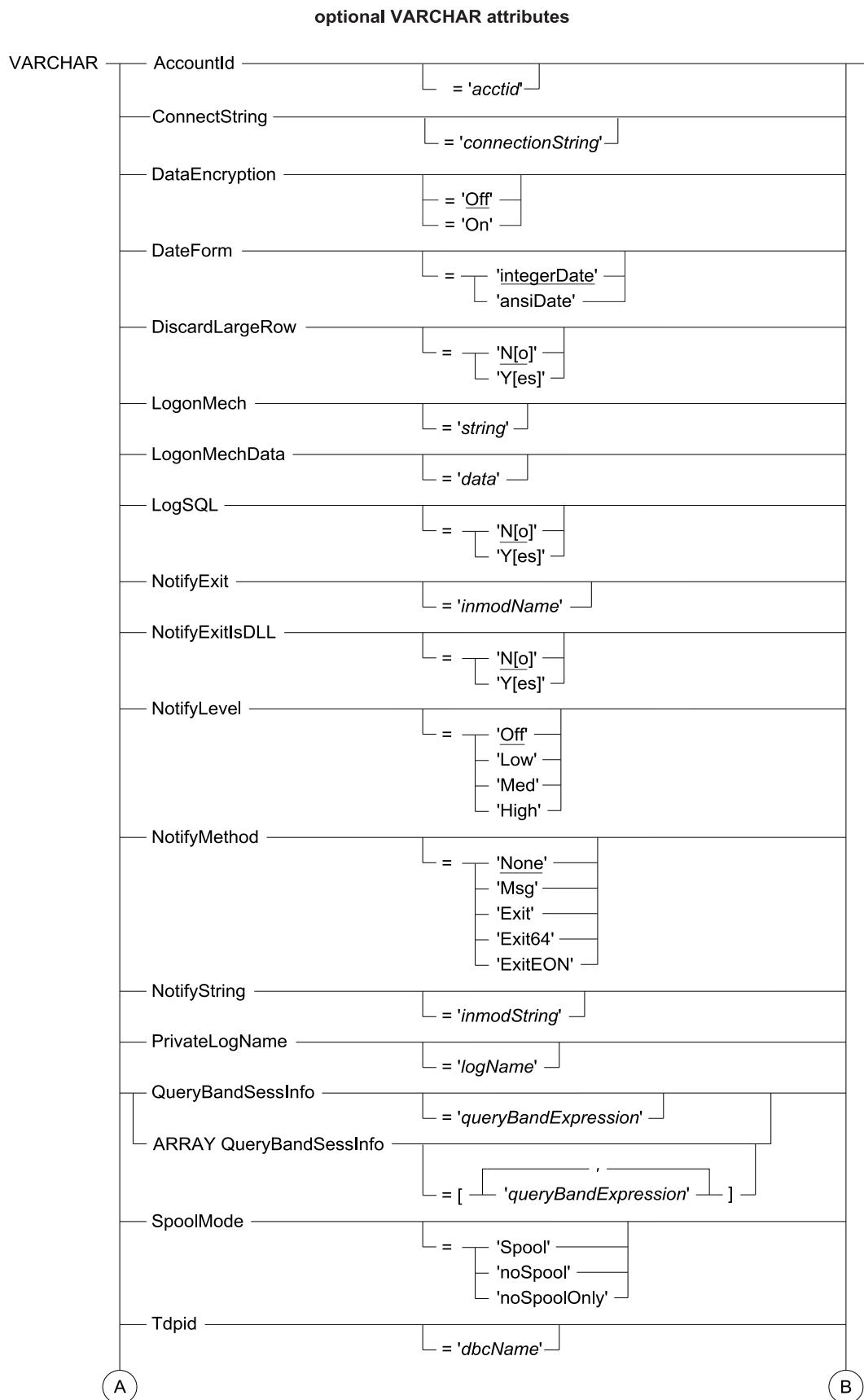
Specification	Description
TYPE	Operator type. Always EXPORT for the Export operator.
UserName	Operator attribute providing the name of the user for the Export operator logon sessions.
UserPassword	Operator attribute providing the password associated with the user name.
SelectStmt	Operator attribute providing the Teradata SQL SELECT statement for the Export operator job.

## Required and Optional Attributes

Use the attribute definition list syntax in the Teradata PT DEFINE OPERATOR statement to declare the required and optional attribute values for the Export operator.



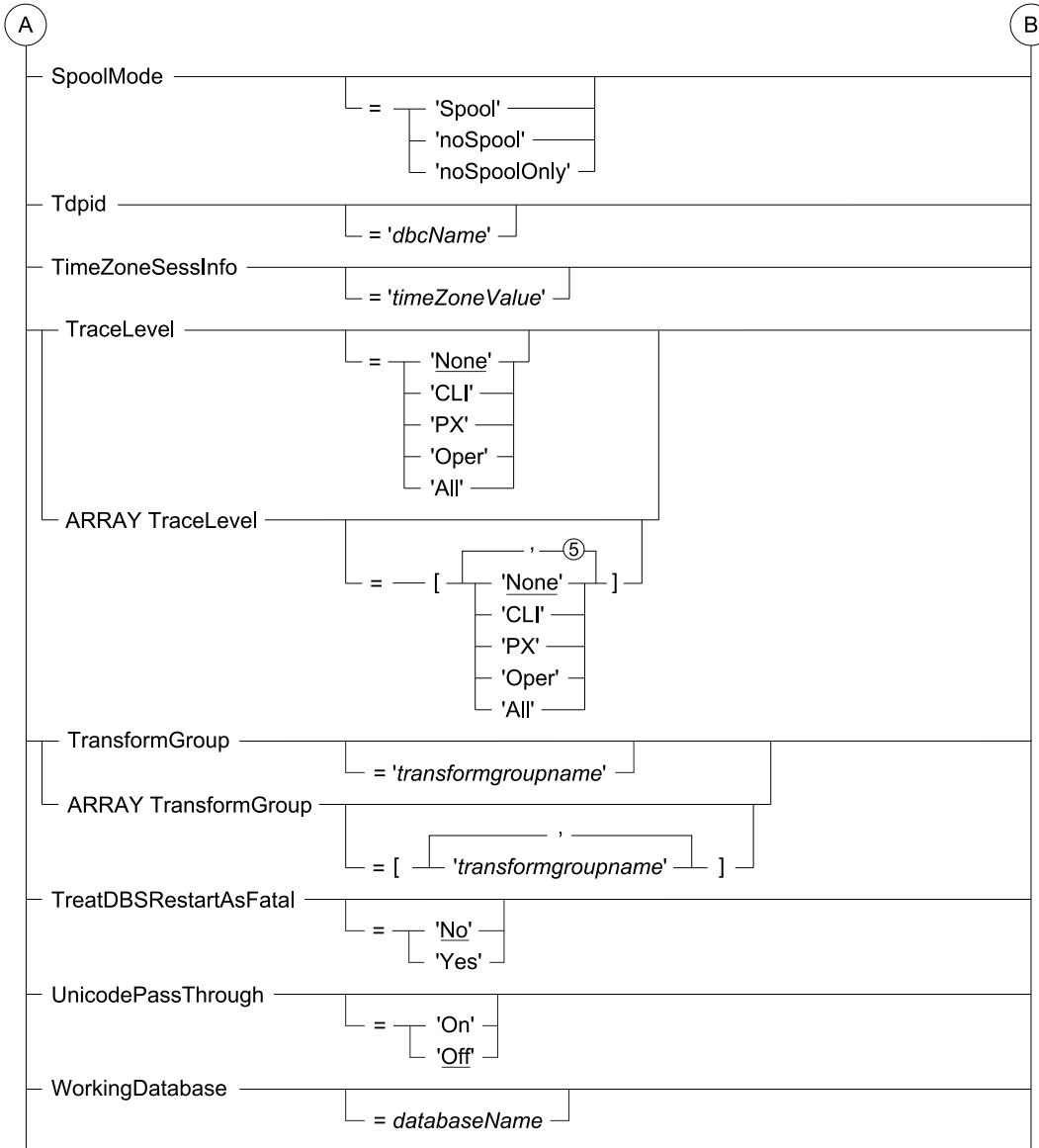




A

B

## optional VARCHAR attributes (continued)



where:

#### Export Operator Attribute Definitions

Syntax Element	Description
AccountId = 'acctId'	Optional attribute that specifies the account associated with the specified user name. If omitted, it defaults to the account identifier of the immediate owner database.
BlockSize = bytes	Optional attribute that specifies the block size to use when returning data to the client. The minimum is 256 bytes. The default is 1048472 bytes. The maximum is 16775168 bytes.

Syntax Element	Description
	<p><b>Note:</b></p> <p>The BlockSize value cannot be larger than the message size supported by the database. If the supplied block size is too large, the operator scales it back to the maximum allowed block size and the job continues.</p>
ConnectionString = ' <i>connectionString</i> '	<p>Optional attribute that specifies the connection string. The connection string will be passed to CLI. CLI will validate the connection string.</p> <p>For information on connection string, see <i>Teradata® Call-Level Interface Version 2 Reference for Workstation-Attached Systems</i>, B035-2418.</p> <p><b>Note:</b></p> <p>The TPT Connection String feature is available on all platforms, except z/OS.</p>
DataEncryption = ' <i>option</i> '	<p>Optional attribute that enables full security encryption of SQL requests, responses, and data.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>• 'On' = all SQL requests, responses, and data are encrypted.</li> <li>• 'Off' = no encryption occurs (default).</li> </ul>
DateForm = ' <i>option</i> '	<p>Optional attribute that specifies the DATE data type for the Export operator job. Valid values are:</p> <ul style="list-style-type: none"> <li>• 'integerDate' = the integer DATE data type (default)</li> <li>• 'ansiDate' = the ANSI fixed-length CHAR(10) DATE data type</li> </ul>
DiscardLargeRow = ' <i>option</i> '	<p>Optional attribute that tells the operator whether to discard large rows (greater than 64K), because the consumer operator cannot handle large rows.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>• 'Yes' or 'Y' = Tells the operator to discard the large rows and continue the job.</li> <li>• 'No' or 'N' = Tells the operator to terminate the job when a large row is encountered (default).</li> </ul> <p><b>Note:</b></p> <p>A consumer operator cannot handle large rows when the consumer operator is talking to a database that does not support a row size greater than 64K.</p> <div style="background-color: #0072bc; color: white; padding: 5px; text-align: center;">  <b>NOTICE</b> </div> <p>When you enable this option and the consumer operator cannot handle large rows, the operator discards the large rows. The discarded rows will not be saved.</p>

Syntax Element	Description
	<p><b>Note:</b></p> <ul style="list-style-type: none"> <li>This option applies only for the FastExport NoSpool mode, because the operator does not know ahead of time whether any rows will be too large for the consumer operator.</li> <li>This option does not apply for the FastExport Spool mode, because the operator knows ahead of time whether any rows will be too large for the consumer operator. When the operator knows ahead of time, the operator will terminate the job right away where there is 1 or more large rows.</li> </ul>
LogonMech = 'string'	<p>Optional attribute that specifies which logon mechanism to use.</p> <p><b>Note:</b></p> <p>Specification of this attribute may be required for some authentication methods.</p> <p>The job terminates if the attribute exceeds 8 bytes.</p> <p>For information on specification requirements for LogonMech "Logon Security" in <i>Teradata® Parallel Transporter User Guide</i>, B035-2445.</p>
LogonMechData = 'data'	<p>Optional attribute that passes along additional logon data.</p> <p><b>Note:</b></p> <p>Specification of this attribute is required for some external authentication methods.</p> <p>For information on specification requirements for LogonMechData "Logon Security" in <i>Teradata® Parallel Transporter User Guide</i>, B035-2445.</p>
LogSQL = 'option'	<p>Optional attribute that controls how much of the job's SQL to enter into the log.</p> <p>Valid options are:</p> <ul style="list-style-type: none"> <li>'Yes' = output the full SQL to the log. The maximum length is 1M.</li> <li>'No' = do not output SQL to the log.</li> <li>No value or attribute omitted = accept the pre-defined limit, which displays up to 32K of SQL if all the SQL is less than 32K. If the SQL to be logged exceeds 32K, truncate the display to the first 32K bytes.</li> </ul>
MaxDecimalDigits = <i>digits</i>	<p>Optional attribute that restricts the number of digits in the DECIMAL data type that can be exported.</p> <p>Valid values are from 1 to 38.</p> <p>The default value is 38.</p> <p>If the attribute has a value of <math>q</math> where <math>38 \geq q \geq 1</math>, then any returned DECIMAL (<math>n[m]</math>) data item with <math>n &gt; q</math> is implicitly CAST to DECIMAL; overflows are handled the same as an explicit CAST.</p> <p>On mainframe-attached platforms, it is recommended that the MaxDecimalDigits attribute not exceed 31 to avoid representations that exceed the capacity of the native instruction set.</p>
MaxSessions = <i>maxSessions</i>	Optional attribute specifying the maximum number of Export sessions to log on.

Syntax Element	Description
	<p>The MaxSessions value must be greater than 0. Specifying a value less than 1 terminates the job.</p> <p>The default is one session per available AMP. The maximum value cannot exceed the number of available AMPs.</p>
MinSessions = <i>minSessions</i>	<p>Optional attribute that specifies the minimum number of sessions required for the Export operator job to continue.</p> <p>The MinSessions value must be greater than 0 and less than or equal to the maximum number of Export operator sessions. The default is one session.</p>
NotifyExit = ' <i>inmodName</i> '	<p>Optional attribute that specifies the name of the user-defined notify exit routine. If no value is supplied, the following default name is used:</p> <ul style="list-style-type: none"> <li>• libnotifyext.dll for Windows platform</li> <li>• libnotifyext.dylib for Apple macOS platforms</li> <li>• libnotifyext.so for all other UNIX platforms</li> <li>• NOTIFYEXT for z/OS platform</li> </ul> <p>See <a href="#">Restricted Words</a> for information about providing your own notify exit routine.</p>
NotifyLevel = ' <i>notifyLevel</i> '	<p>Optional attribute that indicates the level at which certain events are reported. Valid values are:</p> <ul style="list-style-type: none"> <li>• 'Off' = no notification of events is provided (default)</li> <li>• 'Low' = 'Yes' in the Low Notification Level column</li> <li>• 'Med' = 'Yes' in the Medium Notification Level column</li> <li>• 'High' = 'Yes' in the High Notification Level column</li> </ul>
NotifyMethod = ' <i>notifyMethod</i> '	<p>Optional attribute that indicates the method to be used for reporting events. The methods are:</p> <ul style="list-style-type: none"> <li>• 'None' = no event logging is done (default).</li> <li>• 'Msg' = sends the events to a log.</li> </ul> <p>On Windows, the events are sent to the EventLog that can be viewed using the Event Viewer. The messages are sent to the Application log.</p> <p>On Solaris, AIX, and Linux platforms, the destination of the events is dependent upon the setting specified in the /etc/syslog.conf file.</p> <p>On SLES11, the destination of the events is dependent upon the setting specified in the /etc/syslog-ng.conf file.</p> <p>On z/OS systems, events are sent to the job log.</p> <ul style="list-style-type: none"> <li>• 'Exit' = sends the events to a user-defined notify exit routine. Row count information is in 4-byte unsigned integer values.</li> <li>• 'Exit64' = sends the events to a user-defined notify exit routine. Row count information is in 8-byte unsigned integer values for the following events: NXEventStmtFetchEnd64 NXEventReqFetchEnd64 NXEventExportEnd64</li> <li>• 'ExitEON' = sends the events to a user-defined notify exit routine. Complete Teradata object names are passed to the notify exit routine for the NXEventInitializeEON event. In addition, ExitEON sends row count information in 8-byte unsigned integer values.</li> </ul>

Syntax Element	Description
NotifyString = ' <i>notifyString</i> '	<p>Optional attribute that provides a user-defined string to precede all messages sent to the system log. This string is also sent to the user-defined notify exit routine. The maximum length of the string is:</p> <ul style="list-style-type: none"> <li>• 80 bytes, if the NotifyMethod is 'Exit'</li> <li>• 16 bytes, if NotifyMethod is 'Msg'</li> </ul>
OutLimit= <i>maxRecords</i>	<p>Optional attribute that specifies the maximum number of records processed by each instance of the operator.</p> <pre>INTEGER OutLimit = 1000</pre> <ul style="list-style-type: none"> <li>• If specified, OutLimit value must be greater than 0.</li> <li>• If OutLimit is not specified, the number of records processed is not limited.</li> <li>• OutLimit specification applies to each instance of the Export operator and not to the total job.</li> </ul>
PrivateLogName = ' <i>logName</i> '	<p>Optional attribute that specifies the name of a log that is maintained by the Teradata PT Logger inside the public log. The private log contains all the output provided by the Export operator.</p> <p>The private log can be viewed using the tlogview command as follows, where <i>jobid</i> is the Teradata PT job name and <i>privateLogName</i> is the value for the Export operator PrivateLogName attribute:</p> <pre>tlogview -j <i>jobid</i> -f <i>privatelogname</i></pre> <p>If the private log is not specified, all the output is stored in the public log. For more information about the tlogview command, see <a href="#">Teradata PT Utility Commands</a>.</p>
QueryBandSessInfo = ' <i>queryBandExpression</i> '	<p>Optional attribute that specifies the Query Band for the duration of the job sessions.</p> <p><i>queryBandExpression</i> is a set of name=value pairs, separated by a semicolon and ending with a semicolon. The user defines the Query Band expression, which will look similar to the following example:</p> <pre>'org=Finance;load=daily;location=west;'</pre> <p>QueryBandSessInfo may also be specified as an ARRAY attribute. For information on the rules for creating a Query Band expression, see <a href="#">Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</a>, B035-1144 and <a href="#">Teradata Vantage™ - SQL Data Definition Language Detailed Topics</a>, B035-1184.</p> <p>The value of the QueryBandSessInfo attribute is displayed in the Export operator private log.</p> <p>Use of the QueryBandSessInfo attribute is subject to the following rules:</p> <ul style="list-style-type: none"> <li>• By default, Query Band is off until a valid value appears for the QueryBandSessInfo attribute.</li> <li>• If the QueryBandSessInfo attribute contains a value, the Export operator constructs the necessary SET QUERY BAND SQL and issues it as part of the Load operator SQL sessions to communicate the request to the database.</li> </ul>

Syntax Element	Description
	<ul style="list-style-type: none"> <li>The Export operator does not check the Query Band expression, but passes the expression to the database as is.</li> <li>If the version of the database against which the job is being run does not support the Query Band feature, no Query Banding will take place. However, the operator will ignore the error and run the rest of the job.</li> <li>If there is a syntax error in the Query Band expression, the database returns an error. The Export operator will then terminate the job and report the error to the user.</li> </ul>
RoleName - ' <i>role name</i> '	<p>Optional attribute that implements security in a database environment. The operator will prepend the value with "SET ROLE ". The syntax will be sent to the database as follows:</p> <pre>SET ROLE &lt;role name&gt;;</pre> <p>For example:</p> <pre>SET ROLE All;</pre> <p>For details of SET ROLE command use, see <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i>, B035-1144.</p> <p>The VARCHAR ARRAY can specify more than one value, for example:</p> <pre>VARCHAR ARRAY RoleName = ['role name1', 'role name2'],</pre> <p>The operator will send the request to the database on the SQL session after the session is connected.</p> <p>The operator does not send the request on the FastExport protocol sessions, because the database does not allow the request to be sent on the FastExport protocol sessions.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>The operator does not validate the value for this attribute. The operator passes the value to the database as is. The database will validate the value. The operator will terminate the job with an error when the validation fails.</li> <li>C-style comments are allowed in the value and will be passed to the database.</li> <li>ANSI-style comments are not supported in the value. The operator can terminate the job with a syntax error when the value contains an ANSI-style comment.</li> <li>A semicolon is not allowed in the value, because the operator allows only a single statement per request. The operator will terminate the job with an operator error when the value contains a semicolon.</li> </ul>
SelectStmt = 'SELECT <i>statements</i> ,'	<p>Required attribute that specifies the SQL SELECT statement (or statements).</p> <p>Refer to the SQL reference for your operating system and enter one or more valid SQL SELECT statements for the SelectStmt attribute.</p> <p>The statement, or statements, must be enclosed in single quotes and each statement must be terminated with a semicolon.</p>
SpoolMode = ' <i>option</i> '	<p>Optional attribute that specifies which spooling mode to use for the answer set.</p> <ul style="list-style-type: none"> <li>'Spool' tells the Export operator to spool the answer set. This is the default.</li> </ul>

Syntax Element	Description
	<ul style="list-style-type: none"> <li>'NoSpool' tells the Export operator to try using the NoSpool method. If NoSpool is not supported, then use Spool.</li> <li>'NoSpoolOnly' tells the Export operator to use the NoSpool method only. If NoSpool is not supported, the job terminates with an error.</li> </ul>
VARCHAR TASMFASTFAIL = 'value'	<p>Optional attribute that enables FASTFAIL. Valid values are:</p> <ul style="list-style-type: none"> <li>'Yes' or 'Y' = Enable FastFail feature. The job will terminate gracefully when it is supposed to be held by the database.</li> <li>'No' or 'N' = FastFail feature is not enabled (default). The job will appear to hang if the TASM rules dictate that a job should be held for any reason.</li> </ul>
TdplId = 'dbcName'	<p>Optional attribute that specifies the name of the database machine (non-mainframe platforms) or TDP (mainframe platforms) for the Export operator job.</p> <p>The <i>dbcName</i> can be up to 256 characters and can be a domain server name.</p> <p>The <i>dbcName</i> is the name of the host entered in the network hosts file. If you do not specify the value for the TdplId attribute, the operator uses the default TdpID established for the user by the system administrator.</p> <p><b>Note:</b> On a mainframe, a single-character TdplId is supported. When only one character is specified, it is assumed to be an abbreviation for a four-character TdplId that begins with <i>TDP</i>.</p>
TenacityHours = <i>hours</i>	<p>Optional attribute that specifies the number of hours that the Export operator continues trying to log on when the maximum number of Update/Export/Load jobs are already running on the database.</p> <p>The default value is 4 hours. To enable the tenacity feature, <i>hours</i> must be greater than 0. Specifying a value of 0 will disable the tenacity feature. Specifying a value of less than 0 terminates the job.</p>
TenacitySleep = <i>minutes</i>	<p>Optional attribute that specifies the number of minutes that the Export operator pauses before retrying a logon operation when the maximum number of Update/Export/Load operator jobs are already running on the database.</p> <p>The minutes value must be greater than 0. If you specify a value less than 1, the Export operator responds with an error message and terminates the job. The default is 6 minutes.</p>
TimeZoneSessInfo = ' <i>timeZoneValue</i> '	<p>Optional attribute that allows you to change the default time zone displacement for the duration of the operator's job session.</p> <p>When you provide a value for this attribute, the operator will build the "SET TIME ZONE &lt;<i>timeZoneValue</i>&gt;;" SQL request.</p> <p>The operator will send the request to the database on the SQL session after the session is connected.</p> <p>The operator does not send the request on the FastExport protocol sessions, because the database does not allow the request to be sent on the FastExport protocol sessions.</p> <p>Here are some examples:</p>

Syntax Element	Description
	<ul style="list-style-type: none"> <li><b>Example 1:</b> This example sets the session default time zone displacement to LOCAL, which is the system default time zone:  <code>VARCHAR TimeZoneSessInfo = 'LOCAL'</code></li> <li><b>Example 2:</b> This example sets the session default time zone displacement to USER, which is the default time zone for the logged on user:  <code>VARCHAR TimeZoneSessInfo = 'USER'</code></li> <li><b>Example 3:</b> This example sets the session default time zone displacement to a simple constant time zone string expression:  <code>VARCHAR TimeZoneSessInfo = '''America Pacific'''</code></li> </ul> <p><b>Note:</b>  Any single quote character ('') inside the value must be entered as two consecutive single quote characters in a TPT job script. This ensures the correct value will be sent to the database.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>The operator does not validate the value for this attribute. The operator passes the value to the database as is. The database will validate the value. The operator will terminate the job with an error when the validation fails.</li> <li>C-style comments are allowed in the value and will be passed to the database.</li> <li>ANSI-style comments are not supported in the value. The operator will terminate the job with a syntax error when the value contains an ANSI-style comment.</li> <li>A semicolon is not allowed in the value, because the operator allows only a single statement in the "SET TIME ZONE SQL" request. The operator will terminate the job with an operator error when the value contains a semicolon.</li> </ul> <p>For more information on SET TIME ZONE SQL, see <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i>, B035-1144.</p>
TraceLevel = 'level'	<p>Optional attribute that specifies the types of diagnostic messages that are written by each instance of the operator to the public log (or private log, if one is specified using the PrivateLogName attribute). The diagnostic trace function provides more detailed information in the log file to aid in problem tracking and diagnosis. The trace levels are:</p> <ul style="list-style-type: none"> <li>'None' = TraceLevel turned off (default).</li> <li>'CLI' = enables the tracing function for CLI-related activities (interaction with the database)</li> <li>'PX' = enables the tracing function for activities related to the Teradata PT infrastructure</li> <li>'Oper' = enables the tracing function for operator-specific activities</li> <li>'Notify' = enables the tracing function for activities related to the Notify feature</li> <li>'All' = enables tracing for all the previous activities</li> </ul> <p>The VARCHAR ARRAY can specify more than one value, for example:</p>

Syntax Element	Description
	<pre>VARCHAR TraceLevel = 'CLI' VARCHAR TraceLevel = 'OPER' VARCHAR ARRAY TraceLevel = [ 'CLI' ] VARCHAR ARRAY TraceLevel = [ 'CLI', 'OPER' ]</pre> <p><b>Note:</b> The TraceLevel attribute is provided only as a diagnostic aid. The amount and type of additional information provided by this attribute will change to meet evolving needs from release to release.</p>
TransformGroup = 'transformgroupname'	<p>Optional attribute that supports changing the active transform for Teradata Complex Data Types (CDTs). The value is the <i>&lt;udt name&gt; &lt;transform group name&gt;</i>, and the operator will prepend the hardcoded string "SET TRANSFORM GROUP FOR TYPE ". The syntax sent to RDBMS is as follows:</p> <pre>SET TRANSFORM GROUP FOR TYPE &lt;udt name&gt; &lt;transform group name&gt;;</pre> <p>For example:</p> <pre>"SET TRANSFORM GROUP FOR TYPE JSON CHARACTER SET LATIN TD_JSON_VARCHAR;"</pre> <p>The VARCHAR ARRAY can specify more than one value, for example:</p> <pre>VARCHAR ARRAY TransformGroup = [ 'JSON CHARACTER SET LATIN TD_JSON_VARCHAR',                                 'ST_Geometry TD_Geo_VARCHAR' ],</pre> <p>The operator will send the request to the database on the SQL session after the session is connected.</p> <p>The operator does not send the request on the FastExport protocol sessions, because the database does not allow the request to be sent on the FastExport protocol sessions.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>The operator does not validate the value for this attribute. The operator passes the value to the database as is. The database will validate the value. The operator will terminate the job with an error when the validation fails.</li> <li>C-style comments are allowed in the value and will be passed to the database.</li> <li>ANSI-style comments are not supported in the value. The operator can terminate the job with a syntax error when the value contains an ANSI-style comment.</li> <li>A semicolon is not allowed in the value, because the operator allows only a single statement per request. The operator will terminate the job with an operator error when the value contains a semicolon.</li> </ul>
TreatDBSRestartAsFatal= 'option'	<p>Optional attribute that tells the operator whether to terminate the job when a database restart occurs.</p> <p>The TreatDBSRestartAsFatal values are:</p> <ul style="list-style-type: none"> <li>'No' ('N') = The operator will not terminate if a database restart occurs (default). The database restart will be treated as a retryable one.</li> </ul>

Syntax Element	Description
	<ul style="list-style-type: none"> <li>• 'Yes' ('Y') = The operator will terminate if a database restart occurs.</li> </ul>
UnicodePassThrough = 'value'	<p>Optional attribute that tells the operator to enable or disable the Unicode Pass Through feature.</p> <p>Valid values:</p> <ul style="list-style-type: none"> <li>• 'On' = Enable the Unicode Pass Through feature in the operator.</li> <li>• 'Off' = (Default) Disable the Unicode Pass Through feature in the operator.</li> </ul> <p><b>Note:</b></p> <p>When a TPT job is using the UTF8 or UTF16 session character set, the UnicodePassThrough attribute can be set to 'On' to allow the operator to export data with Unicode pass through characters.</p>
UserName = 'userId'	<p>Attribute that specifies the user name on the database.</p> <p><b>Note:</b></p> <p>Use of this attribute is not compatible with some external authentication logon methods. For more information on username specification requirements, see "Logon Security" in <i>Teradata® Parallel Transporter User Guide</i>, B035-2445.</p>
UserPassword = 'password'	<p>Attribute that specifies the password associated with the database user name.</p> <p><b>Note:</b></p> <p>Use of this attribute is not compatible with some external authentication logon methods. For more information on password specification requirements, see "Logon Security" in <i>Teradata® Parallel Transporter User Guide</i>, B035-2445.</p>
VARCHAR	Keyword that specifies VARCHAR as the data type of the defined attribute.
WorkingDatabase = 'databaseName'	<p>Optional attribute that specifies a database other than the logon database as the default database.</p> <p>The name of the database that is specified with this attribute is used in the SQL DATABASE statement that is sent by the operator immediately after connecting the two SQL sessions.</p> <p>If WorkingDatabase is not specified, the default database associated with the logged on user is assumed for all unqualified table names.</p>

## Usage Notes

### SelectStmt

SelectStmt (SELECT statement) is the required attribute that the Export operator uses to perform data selection from the database tables. Multiple parallel instances of the Export operator and multiple sessions within each instance can improve the performance of an export.

A Select request within an Export script can have multiple SELECT statements. A SELECT statement can be optionally preceded by a LOCKING modifier.

However, Export SELECT requests cannot:

- Specify a USING modifier.
- Access non-data tables, such as SELECT DATE or SELECT USER.
- Be satisfied by one or two AMPs, such as SELECT statement which accesses rows based on the primary index or unique secondary index of a table.
- Contain character large object (CLOB) or binary large object (BLOB) data types.
- Contain JSON (JavaScript Object Notation) data type.
- Contain XML data type.

The following table describes types of SELECT requests.

Type of SELECT Request	Result
Contains multiple SELECT statements	The database might execute the requests in parallel, but responses are still returned in the order of the requests, for the first statement first, then for the second, and so on. If the structure of response rows differs, an error results and the job terminates.
Uses a LOCKING modifier	<p>The specified lock remains in effect during the execution of all statements within the request that contains the modifier. The database does the following:</p> <ul style="list-style-type: none"> <li>Implements all resource locks for the entire request before executing any of the statements in the request.</li> <li>Maintains the locks until all the response data for the request is moved to spool tables.</li> </ul> <p>Following is a valid SELECT request using the LOCKING modifier:</p> <pre>LOCKING TABLE MYTABLE FOR ACCESS SELECT COL1, COL2 FROM MYTABLE;</pre> <p>Note that the LOCKING modifier can precede the SELECT statement.</p>
Uses an ORDER BY clause	<p>You can specify one or multiple Export instances. If you specify multiple instances, only instance 1 will connect sessions to preserve the exported rows in order. The other instances will not connect any sessions and will not export any data.</p> <p>Following is a valid ORDER BY clause:</p> <pre>SELECT COL1, COL2 FROM MYTABLE ORDER BY COL1;</pre>

## Multiple SELECT Statements

A single Export operator can read data from multiple tables if their schemas are the same.

For example, if Table1 and Table2 have the same schema, use the following SELECT statement in the attribute section of the Export operator definition:

```
VARCHAR SelectStmt='SELECT * FROM Table1;SELECT * FROM Table2'
```

If the job contains multiple SELECT statements and the job uses multiple instances of the Export operator, only the first instance of the Export operator logs on the special session(s). This is necessary to preserve the exported records in statement order.

## NoSpool Mode

The NoSpool mode exports the contents of a table as fast as possible without reading the table into a spool file or distributing the file to all AMPs before extracting it. The spooling options follow:

- (Default) SPOOL the data.
- Use the NOSPOOLONLY mode, but return an error if NOSPOOL is not supported.
- Use the NOSPOOL mode when possible; otherwise spool the data in the database.

## Limitations and Functionality

- NOSPOOL mode applies only to simple SELECT statements. The following are not supported:
  - Access to nondata tables, such as SELECT DATE or SELECT USER
  - USING modifier; instead, define restraint parameters by using a FastExport IMPORT command with supporting FIELD and FILLER commands
  - Contains a SORT (ORDER BY), HAVING, or WITH clauses
  - Joins
  - Aggregations (Explain shows SUM step)
  - TABLE functions
  - Ordered-analytic (OLAP) functions
  - Multiple SELECT statements or multistatement requests
  - Statements with zero or more than one, retrieve or sampling step
- NOSPOOL mode only retrieves data from a single table, but the SELECT statement can be selective about which columns are exported and can constrain the job to a subset of rows.
- Scalar expressions/functions are allowed.
- The Sample and partition eliminating constraints are supported.
- The Activity Count returned for a regular spooled job indicates the number of affected blocks; however, for non-spoiled jobs, the number of blocks is unknown, so the response message contains ActivityType (instead of Activity Count) to indicate the NOSPOOL process.

## Disadvantages of the NOSPOOL Mode

- Locks are maintained during the entire export process.
- Data conversion errors previously detected during the spooling phase will not be detected until the block is read, which could occur any time during the export.
- Row order (because of the absence of the ORDER BY clause) may or may not be consistent between runs; therefore, NOSPOOL mode offers no guarantee of consistency.

See information on Spool/NoSpool mode, *Teradata FastExport Reference* (B035-2410).

## Restrictions and Limitations

This section describes the restrictions and limitations when using the Export operator.

### SELECT Statements

Only the Teradata SQL SELECT statements are supported in the Export operator's SelectStmt attribute.

The SELECT statements in the Export operator's SelectStmt attribute cannot have any of the following characteristics:

- Contain a USING modifier
- Access nondata tables, such as SELECT DATE or SELECT USER

All SELECT statements in the Export operator's SelectStmt attribute are treated as a single multi-statement request to be sent to the database.

### Data Types

The job cannot export these data types:

- LOB
- JSON
- XML
- ST\_Geometry

If the ST\_Geometry data type is casted as VARCHAR/VARBYTE, then the data can be exported.

UDT data can be exported in its external type format.

### Notes

- The Export operator can export rows from multiple tables and views as long as the result sets adhere to the same schema.

- For more information about restrictions on using the NoSpool mode, see "Limitations and Functionality" in [NoSpool Mode](#).
- The Export operator is not restartable if the data export has started and the data export has not completed.
- The Export operator requires one database load slot.

## Job Options

### Limits on Export Jobs

The Export operator requires one database load slot.

The number of concurrent load slots is, however, configurable in the database environment using the same MaxLoadTasks field control used by FastLoad, FastExport, and MultiLoad. For example, one Teradata PT Export Load job equates to one FastExport job in the count for MaxLoadTasks.

For more information, see "DBS Control Utilities" in *Teradata Vantage™ - Database Utilities*, B035-1102.

### Limiting Output

Use the OUTLIMIT attribute to limit the number of rows that an export job returns. OUTLIMIT expects a number greater than zero as input. Specifying an invalid number terminates the export operator and returns an error message. If OUTLIMIT is not specified, the export operator returns all records. OUTLIMIT controls the number of rows exported for each instance.

You can also use one of the following techniques to limit record output:

- The WHERE clause can limit the number of rows that are exported by specifying conditions that must be met.
- The SAMPLE function is an SQL function that can limit the number of random rows returned. For more information, see *Teradata Vantage™ - SQL Data Manipulation Language*, B035-1146.

### Checkpointing and Restarting

The Export operator takes a checkpoint only when all data is sent to the data stream. If a restart occurs, the operator either must send all of the data or none of the data depending on whether the checkpoint has taken place.

- If all the data is sent, then the operator displays the following message and does not resend any of the data:

Restart indicates that this export job completed.

- If all the data is not sent, the operator terminates. Restart the job from the beginning.
- If none of the data is sent, the operator sends the data.

The Export operator does *not* support a user-defined restart log table.

 **NOTICE**

If a checkpoint interval is specified on the `tbuild` command line, the checkpoints incurred between the start of data loading and the end of data loading are ignored by the Export operator.

## Operational Considerations

### Performance

The exporting ability of the Export operator is designed to outperform the SQL Selector operator (similar to BTEQ Export) in transferring large amounts of data from tables.

This is so because, among other factors, the Export operator is multisessioned and able to run in multiple parallel instances. It performs block exports in contrast to the SQL Selector operator, which performs single-session and one-row-at-a-time exports.

Export functions in Teradata PT do not make the database perform faster. Rather, Teradata PT allows for the greater use of multiple parsing engines and AMPs.

### Sessions and Instances

The Export operator will connect one main (control) SQL session in addition to the data sessions. The main SQL session is responsible for executing SQL statements pertaining to utility work.

A minimum and a maximum number of sessions (the session limits) can be specified for the Export operator.

Consider the following usage notes:

- The maximum sessions connected can never exceed the number of available AMPs in the system, even if a larger number is specified.
- The default is one session per available AMP.
- For the `MinSessions` attribute, the minimum specification is one.
- The `MaxSessions` attribute can be set to a number smaller than the number of AMPs on the database server if fewer sessions are suitable for the job.
- Network protocol software might also impose limits on workstation-attached systems.
- Platform limits for maximum sessions per application differ:
  - On a mainframe-attached z/OS client system, use the `TDP SET MAXSESSIONS` command to specify a platform limit.
  - On workstation-attached client systems for UNIX, Linux, and Windows systems, this value is defined in the CLI file, `clispb.dat`, under the `max_num_sess` variable.
  - On mainframe-attached z/OS client systems, this value is defined in the `HSHSPB` parameter under the `IBCSMAX` setting.

The *max\_num\_sess* value in the *clispb.dat* file (or *HSHSPB*) specifies the total number of sessions allowed to be connected by a single application at one time. The *max\_num\_sess* pertains to all sessions connected, both SQL and data loading.

## Multiple Databases

When reading data from multiple databases with different login information, specify values for the *UserName* and *Password* attributes in the *SELECT* portions of the *APPLY* statements that specify each database.

## Exporting VARBYTE Data

The Export operator allows *VARCHAR* and *VARBYTE* data with a length less than or equal to the defined length in the *DEFINE SCHEMA* definition.

# FastExport OUTMOD Adapter Operator

## FastExport OUTMOD Adapter Operator Capabilities

The FastExport OUTMOD Adapter operator is a consumer operator that allows you to use Teradata FastExport OUTMOD routines within the Teradata PT.

The term OUTMOD is an acronym for “*output modification*.” OUTMOD routines are user exit routines the FastExport OUTMOD can call to provide enhanced processing functions on output data records from Teradata PT producer operators before they are sent to the client system.

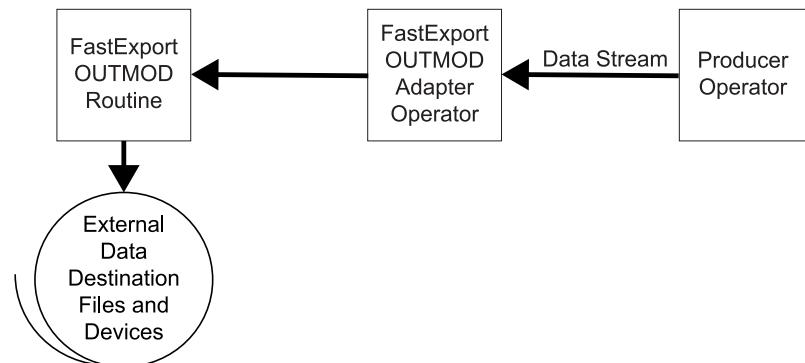
For information on creating and using FastExport OUTMOD Routines *Teradata FastExport Reference* (B035-2410).

**Note:**

The FastExport OUTMOD Adapter operator does not support the OUTMOD routines for any other Teradata standalone load or unload utility.

The following figure shows the FastExport OUTMOD Adapter operator interface.

### FastExport OUTMOD Adapter Operator Interface



**Note:**

The existing TPT function names should not be used in User's OUTMODs. Using the existing TPT function names in User's OUTMODs might cause the normal operation of the TPT job to behave with unexpected results and prevent requests from completing normally.

## Syntax

Use the following specifications and attributes to define the FastExport OUTMOD Adapter operator in a Teradata PT job script.

## Required Specifications

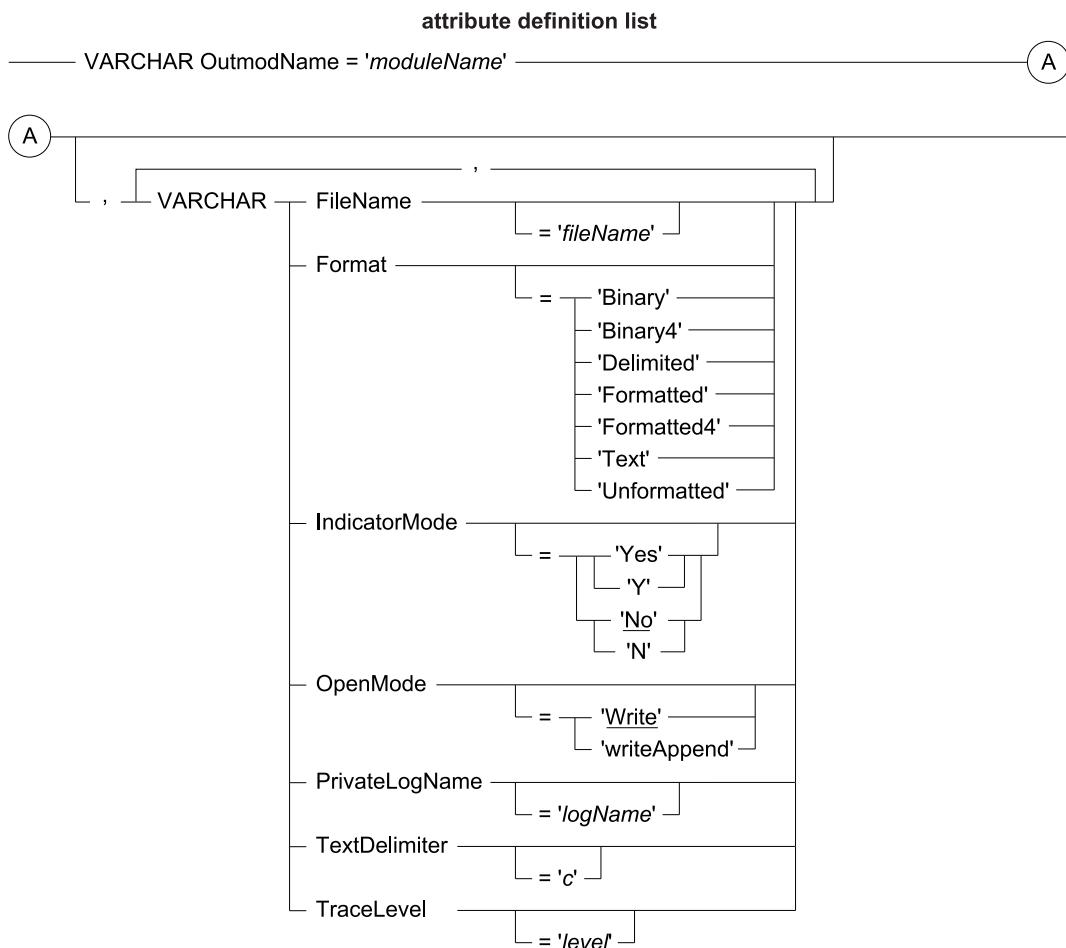
The following specifications are required for this operator.

**Required Syntax for the FastExport OUTMOD Adapter Operator**

Specification	Description
TYPE	Operator type. Always FASTEXPORT OUTMOD for the FastExport OUTMOD Adapter operator as a consumer operator.
OutmodName	Operator attribute providing the name of the OUTMOD routine source file.

## Required and Optional Attributes

Use the attribute definition list syntax in the Teradata PT DEFINE OPERATOR statement to declare the required and optional attribute values for the FastExport OUTMOD Adapter operator.



where:

#### FastExport OUTMOD Adapter Operator Attribute Definitions

Syntax Element	Description
FileName = ' <i>fileName</i> '	Specifies the name of the file. On the UNIX system, limited to 255 bytes.
Format = ' <i>option</i> '	<p>Specifies the logical record format of the file being read.</p> <ul style="list-style-type: none"> <li>• 'Binary' = 2-byte integer, n, followed by n bytes of data. This data format requires rows to be 64KB (64260 data bytes) or smaller. In this format: The data is prefixed by a record-length marker. The record-length marker does not include the length of the marker itself. The record-length is not part of the transmitted data.</li> <li>• 'Binary4' = 4-byte integer, followed by n bytes of data. This data format supports rows up to 1MB (1000000 data bytes) in size. In this format: The data is prefixed by a record-length marker. The record-length marker does not include the length of the marker itself. The record-length is not part of the transmitted data.</li> </ul>

Syntax Element	Description
	<ul style="list-style-type: none"> <li>'Delimited' = in text format with each field separated by a delimiter character. When you specify Delimited format, you can use the optional TextDelimiter attribute to specify the delimiter character. The default is the pipe character (   ). When the assigned value of an operator attribute is 'Delimited' all columns in the schema specified for this operator must be of the VARCHAR data type. Non-VARCHAR specifications will result in an error.</li> <li>'Formatted' = both prefixed by a record-length marker and followed by an end-of-record marker. This data format requires rows to be 64KB (64260 data bytes) or smaller.sw. In this format: The record-length marker does not include the length of the marker itself. Neither the record-length nor the end-of-record marker is part of the transmitted data.</li> <li>'Formatted4' = both prefixed by a 4-byte record-length marker and followed by an end-of-record marker. This data format supports rows up to 1MB (1000000 data bytes) in size. In this format: The record-length marker does not include the length of the marker itself. Neither the record-length nor the end-of-record marker is part of the transmitted data.</li> <li>'Text' = character data separated by an end-of-record (EOR) marker. The EOR marker can be either a single-byte linefeed (X'0A') or a double-byte carriage-return /line-feed pair (X'0D0A'), as defined by the first EOR marker encountered for the first record.</li> <li>'Unformatted' = not formatted. Unformatted data has no record or field delimiters, and is entirely described by the specified Teradata PT schema.</li> </ul>
IndicatorMode = 'mode'	<p>Optional attribute that specifies whether indicator byte(s) is included at the beginning of each record.</p> <ul style="list-style-type: none"> <li>'Yes' (or 'Y') = indicator mode data</li> <li>'No' (or 'N') = nonindicator mode data (default)</li> </ul>
OpenMode = 'mode'	<p>Optional attribute that specifies the read/write access mode.</p> <ul style="list-style-type: none"> <li>'Write' = Write-only access (default)</li> <li>'Write Append' = Write-only access appending to existing file</li> </ul>
OutmodName = 'moduleName'	<p>Required attribute that specifies the name of the FastExport OUTMOD routine to be loaded.</p> <p>The OUTMOD is found in the directory defined in the system-dependent library path. Using the "./" syntax (for example ./f1outmod) indicates the directory in which the job is run.</p>
PrivateLogName = 'logName'	<p>Optional attribute that specifies the name of a log that is maintained by the Teradata PT Logger inside the public log. The private log contains all the output provided by the operator.</p> <p>The private log can be viewed using the tlogview command as follows, where jobId is the Teradata PT job name and privateLogName is the value for the operator's PrivateLogName attribute:</p> <pre>tlogview -j jobId -f privateLogName</pre> <p>If the private log is not specified, all output is stored in the public log.</p>

Syntax Element	Description
	For more information about the tlogview command, see <a href="#">Teradata PT Utility Commands</a> .
TextDelimiter = 'c'	Optional attribute that specifies the character that separates fields in delimited records. The default is the pipe character (   ).
TraceLevel = 'level'	Optional attribute that specifies the types of diagnostic messages that are written by each instance of the operator to the public log (or private log, if one is specified using the PrivateLogName attribute). The diagnostic trace function provides more detailed information in the log file to aid in problem tracking and diagnosis. The trace levels are: <ul style="list-style-type: none"> <li>• 'None' = Enables minimal information such as initialization, access module attach /detach operations, file opening, error messages, and statistics (number of records processed and processor time used). This is the default.</li> <li>• 'Milestones' = enables the trace function only for major events such as initialization, file openings and closings, error conditions, and so on</li> <li>• 'IO_Counts' = enables the trace function for major events plus I/O counts</li> <li>• 'IO_Buffers' = enables the trace function for major events and I/O counts plus I/O buffers</li> <li>• 'All' = enables the trace function for major events and I/O counts and buffers plus function entries</li> </ul> If you use the PrivateLogFile attribute to specify a log file, the TraceLevel value defaults to 'Milestones'. <b>Note:</b> The TraceLevel attribute is provided as a diagnostic aid only. The amount and type of additional information provided by this attribute changes to meet evolving needs from release to release.

## Usage Notes

### Format

The internal format of each output record must conform to the specified schema.

The FastExport OUTMOD Adapter operator requires each field of the output record to be of a specified length. When no length is specified, the operator assumes the conventional default length.

The FastExport OUTMOD Adapter operator supports both fixed- and variable-length output records.

For records with a two-column schema such as:

```
(  
  COL1 INTEGER,  
  COL2 VARCHAR(10));
```

each record comprises:

- A four-byte integer, because 4 is the INTEGER length
- A two-byte length indicator for the VARCHAR field
- The CHAR data of the length indicated in the two-byte length indicator field

The following examples show the byte order when running Teradata PT on an Intel processor.

- If, for example, the integer has a decimal value of 100 and the VARCHAR is ASCII '12345', then the output record appears in hexadecimal notation as:

```
X'64 00 00 00 05 00 31 32 33 34 35'
```

- If the data in the output record is in indicator mode, there is a single preceding indicator byte:

```
X'00 64 00 00 00 05 00 31 32 33 34 35'
```

## Restrictions and Limitations

This section describes the restrictions and limitations when using the FastExport OUTMOD Adapter operator.

### Data Types

The operator cannot process these data types:

- BLOB
- CLOB
- JSON
- XML

## Job Options

The FastExport OUTMOD Adapter operator provides the same functionality and is used in the same circumstances as the DataConnector operator.

## Operational Considerations

The FastExport OUTMOD Adapter provides the same functionality and is used in the same circumstances as the DataConnector operator.

# FastLoad INMOD Adapter Operator

## FastLoad INMOD Adapter Operator Capabilities

The FastLoad INMOD Adapter operator is a producer operator that allows you to use Teradata FastLoad utility INMOD routines within Teradata PT.

The term INMOD is an acronym for “*input modification*.” INMOD routines are user exit routines that FastLoad INMOD adapter can call to provide enhanced processing functions on input data records before they are sent to the database.

You can use the FastLoad INMOD Routine with the FastLoad INMOD Adapter to read and preprocess input data values from files on the client system.

For information on creating and using FastLoad INMOD Routines, see *Teradata® FastLoad Reference*, B035-2411.

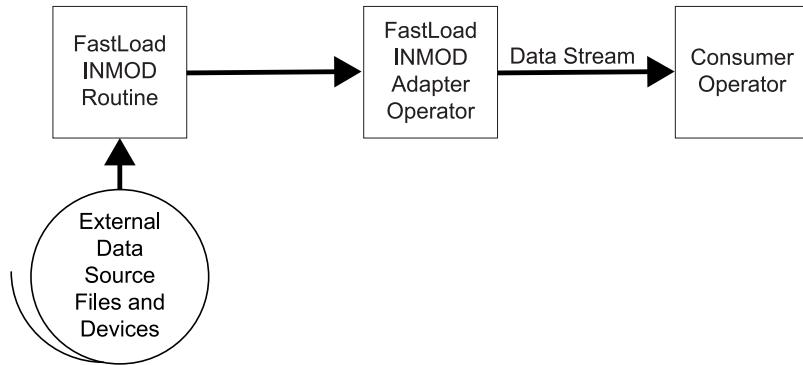
---

### Note:

The FastLoad INMOD Adapter operator does not support the INMOD routines for any other Teradata standalone load or unload utility.

---

The following figure shows the FastLoad INMOD Adapter operator interface.




---

### Note:

The existing TPT function names should not be used in User's INMODs. Using the existing TPT function names in User's INMODs might cause the normal operation of the TPT job to behave with unexpected results and prevent requests from completing normally.

## Syntax

Use the following required and optional specifications to define the FastLoad INMOD Adapter operator in a Teradata PT job script.

## Required Specifications

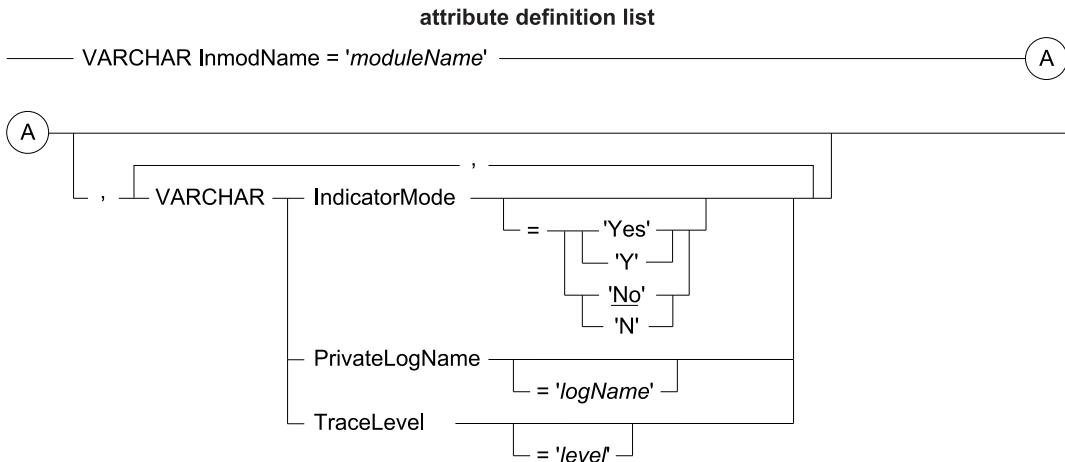
The following specifications are required to define a Load job. See [Required and Optional Attributes](#) for information about defining required and optional operator attributes.

### Required Syntax for the FastLoad INMOD Adapter Operator

Specification	Description
TYPE	Operator type. Always FASTLOAD INMOD for the FastLoad INMOD Adapter operator as a producer operator.
InmodName	Operator attribute providing the name of the INMOD routine source file.

## Required and Optional Attributes

Use the attribute definition list syntax in the Teradata PT DEFINE OPERATOR statement to declare the required and optional attribute values for the FastLoad INMOD Adapter operator.



where:

### FastLoad INMOD Adapter Operator Attribute Descriptions

Syntax Element	Description
IndicatorMode = 'mode'	Optional attribute that specifies whether the indicator byte(s) is included at the beginning of each input record. The values are: <ul style="list-style-type: none"><li>'Yes' (or 'Y') = indicator mode data</li><li>'No' (or 'N') = nonindicator mode data (default)</li></ul>

Syntax Element	Description
InmodName = ' <i>moduleName</i> '	<p>Required attribute that specifies the name of the FastLoad INMOD routine. The INMOD is found in the directory defined in the system-dependent library path. Using the “.” syntax (for example <code>./flinmod</code>) indicates the directory in which the job is run.</p>
PrivateLogName = ' <i>logName</i> '	<p>Optional attribute that specifies the name of a log that is maintained by the Teradata PT Logger inside the public log. The private log contains all of the diagnostic trace messages produced by the operator.</p> <p>The file name is appended with the operator instance number. A “-1” is appended to the log name for instance 1. For example, if <code>PrivateLogName = 'DClog'</code>, then the actual log name for instance 1 is <code>DClog-1</code>. Similarly, for instance 2, it is <code>DClog-2</code>, etc.</p> <p>The private log can be viewed using the <code>tlogview</code> command as follows, where <code>jobId</code> is the Teradata PT job name and <code>privateLogName</code> is the value for the operator’s <code>PrivateLogName</code> attribute:</p> <pre><code>tlogview -j jobId -f privateLogName</code></pre> <p>If the private log is not specified, all of the output is stored in the public log. For more information about the <code>tlogview</code> command, see <a href="#">Teradata PT Utility Commands</a>.</p>
TraceLevel = ' <i>level</i> '	<p>Optional attribute that specifies the types of diagnostic messages that are written by each instance of the operator to the public log (or private log, if one is specified using the <code>PrivateLogName</code> attribute). The diagnostic trace function provides more detailed information in the log file to aid in problem tracking and diagnosis.</p> <p>The trace levels are:</p> <ul style="list-style-type: none"> <li>• 'None' = Enables minimal information such as initialization, access module attach /detach operations, file opening, error messages, and statistics (number of records processed and processor time used). This is the default.</li> <li>• 'Milestones' = enables the trace function only for major events such as initialization, file openings and closings, error conditions, and so on.</li> <li>• 'IO_Counts' = enables the trace function for major events plus I/O counts.</li> <li>• 'IO_Buffers' = enables the trace function for major events and I/O counts plus I/O buffers.</li> <li>• 'All' = enables the trace function for major events and I/O counts and buffers plus function entries.</li> </ul> <p>If you use the <code>PrivateLogFile</code> attribute to specify a log file, and do not specify the <code>TraceLevel</code> attribute, minimal statistics are displayed in the log file, including:</p> <ul style="list-style-type: none"> <li>• Name of files as they are processed</li> <li>• Notice when sending rows begins</li> <li>• Upon completion of each file, the number of rows processed and the CPU time consumed.</li> <li>• Total files processed and CPU time consumed by each instance of the DataConnector operator.</li> </ul> <p>The <code>PrivateLogFile</code> attribute default is used only if a non-zero <code>TraceLevel</code> attribute is specified.</p> <p>If no <code>TraceLevel</code> attribute is specified, or if the specified value is 'None', and the <code>PrivateLogFile</code> attribute is specified, the <code>TraceLevel</code> is set to 'Milestones'.</p>

Syntax Element	Description
	<ol style="list-style-type: none"> <li>1. The TraceLevel attribute is provided as a diagnostic aid only. The amount and type of additional information provided by this attribute will change to meet evolving needs from release to release.</li> <li>2. The recommended TraceLevel value is 'None', which produces NO log file. Specifying any value greater than 'IO_Counts' produces a very large amount of diagnostic information.</li> </ol>
VARCHAR	Keyword specifying VARCHAR as the data type of the defined attribute.

## Usage Notes

### Format

The internal format of each input record with the FastLoad INMOD Adapter operator must conform to the schema.

The FastLoad INMOD Adapter operator requires each field of an input record to a certain length. When no length is specified, the operator assumes the conventional default length.

The FastLoad INMOD Adapter operator supports both fixed- and variable-length input records. For records with a two-column schema such as:

```
(  
  COL1 INTEGER,  
  COL2 VARCHAR(10));
```

Each record must comprise:

- A four-byte integer, because 4 is the INTEGER default,
- A two-byte length indicator for the VARCHAR field, and
- The CHAR data of the length indicated in the two-byte length indicator field.

The following examples show the byte order when Teradata PT is running on an Intel processor.

- For example, if the integer has a decimal value of 100 and the VARCHAR is ASCII '12345', then the input record appears in hexadecimal notation, as:

```
X'64 00 00 00 05 00 31 32 33 34 35'
```

- If the data in the input record is in indicator mode, there is a single preceding indicator byte:

```
X'00 64 00 00 00 05 00 31 32 33 34 35'
```

## Restrictions and Limitations

This section describes the restrictions and limitations when using the FastLoad INMOD Adapter operator.

### Data Types

The operator cannot process these data types:

- BLOB
- CLOB
- JSON
- XML

## Job Options

The FastLoad INMOD Adapter operator provides the same functionality and is used in the same circumstances as the DataConnector operator.

## Operational Considerations

The FastLoad INMOD Adapter operator provides the same functionality and is used in the same circumstances as the DataConnector operator.

# Load Operator

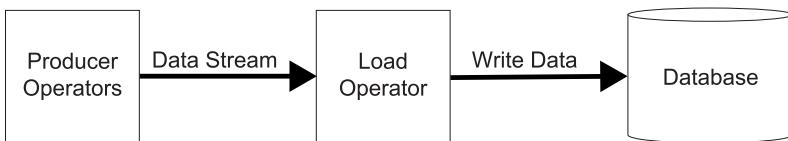
## Load Operator Capabilities

The Load operator, a consumer operator, uses Teradata FastLoad protocol to load a large volume of data at high speed into an empty table in the database.

The Load operator is typically used for initial loading of tables. It inserts data it consumes from data streams into individual rows of a target table.

## Load Operator Interface

The following figure shows the Load operator interface.



## The Load Operator Function in a Teradata PT Job

When you use the Load operator in a Teradata PT job, Teradata PT directs each parallel instance of the Load operator to do the following:

1. Log in to the database, using your user name, password, database name, and account ID information specified in the job script.
2. Load the input data into the Load target table on the database.
3. Log off the database.
4. If the Load operator job is successful, terminate the job and provide information about the job in the log or to the user, such as:
  - Total number of records read and sent to the database
  - Number of errors posted to the error tables
  - Number of inserts applied
  - Number of duplicate rows
  - A status code indicating the success of the job
5. If the job is unsuccessful, terminate the job and provide information about the job so that you can correct any problem and restart the job.

## Load Operation Phases

Load operations have two phases:

- **Acquisition Phase** – Data from the input stream is transmitted to the AMPs. The AMPs 'acquire' the data. The acquisition phase is not complete until all data sources are exhausted and all data rows are on the appropriate AMPs. The data is not yet sorted or blocked, and therefore, is not yet accessible.
- **Application Phase** – Data is sorted, blocked, and put into its final format. All activity in this phase is AMP-local. Data is accessible after the completion of this phase.

## FastLoad Utility and the Load Operator

The following table lists the operating features and capabilities of the Teradata FastLoad utility, and indicates whether they are supported by the Load operator.

**Load Operator Supported Features**

FastLoad Utility Feature	Teradata PT Operator Support
Access Modules	Supported, using the DataConnector operator
ANSI Date	Supported
Checkpoint/Restart	Supported
Character Sets	Supported, using the USING CHARACTER SET clause before the DEFINE JOB statement
Configuration File	Supported, using the <b>tbuild</b> command line job attribute option (-v)
CREATE TABLE Statement	Supported, using the DDL operator
DATABASE Statement	Supported
DELETE Statement	Supported, using the DDL operator
DROP TABLE Statement	Supported, using the DDL operator
Error Limit	Supported
Indicator Mode	Supported, using the DataConnector operator
INMOD Routines	Supported, using the FastLoad INMOD Adapter operator
Maximum/Minimum Sessions	Supported
Nonindicator Mode	Supported, using the DataConnector operator
Notify	Supported
NULLIF Clauses	Supported, but not the XB, XC, and XG data types
Operating System Command	Supported, using the OS Command operator
Record Formats	Supported, using the DataConnector operator

FastLoad Utility Feature	Teradata PT Operator Support
RECORD <i>n</i> THRU <i>m</i>	Supported, in a limited form by the DataConnector operator, allowing you to read in the first "m" rows of a file, effectively allowing "RECORD 1 THRU m"
Show Version Information	Supported
Tenacity	Supported
Wildcard INSERT	Supported

## Syntax

Use the following specifications and attributes to define the Load operator in a Teradata PT job script.

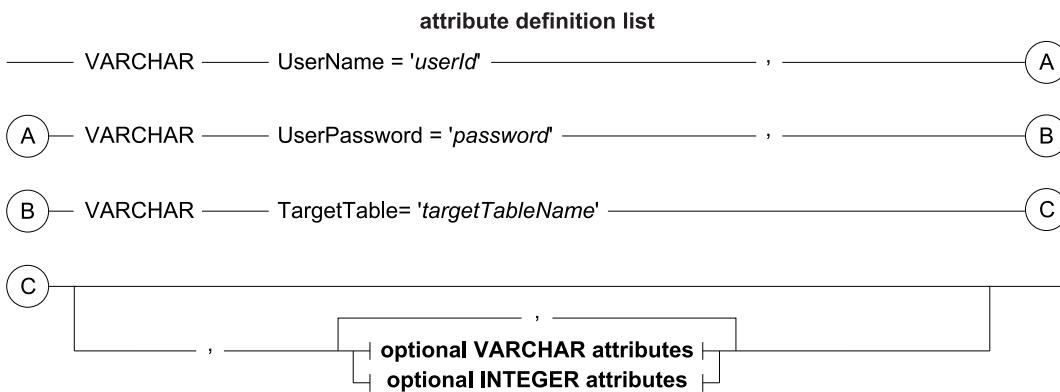
## Required Specifications

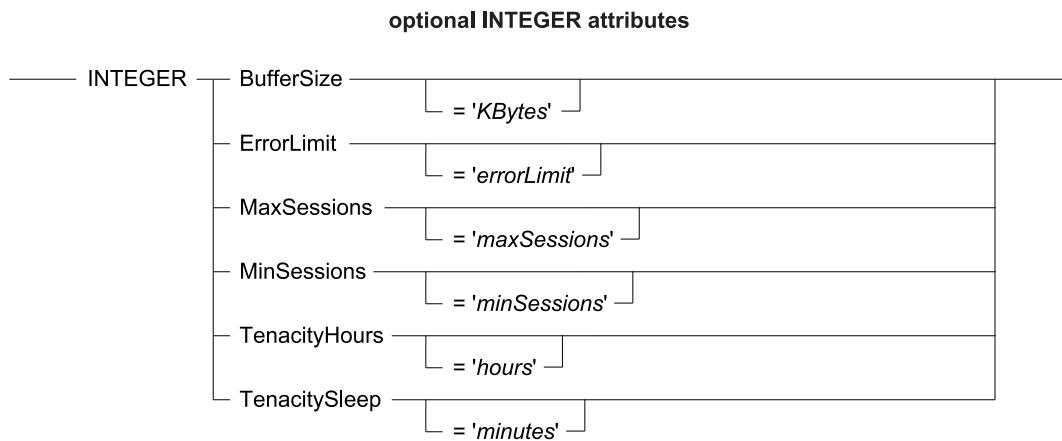
The following specifications begin to define a Load operator.

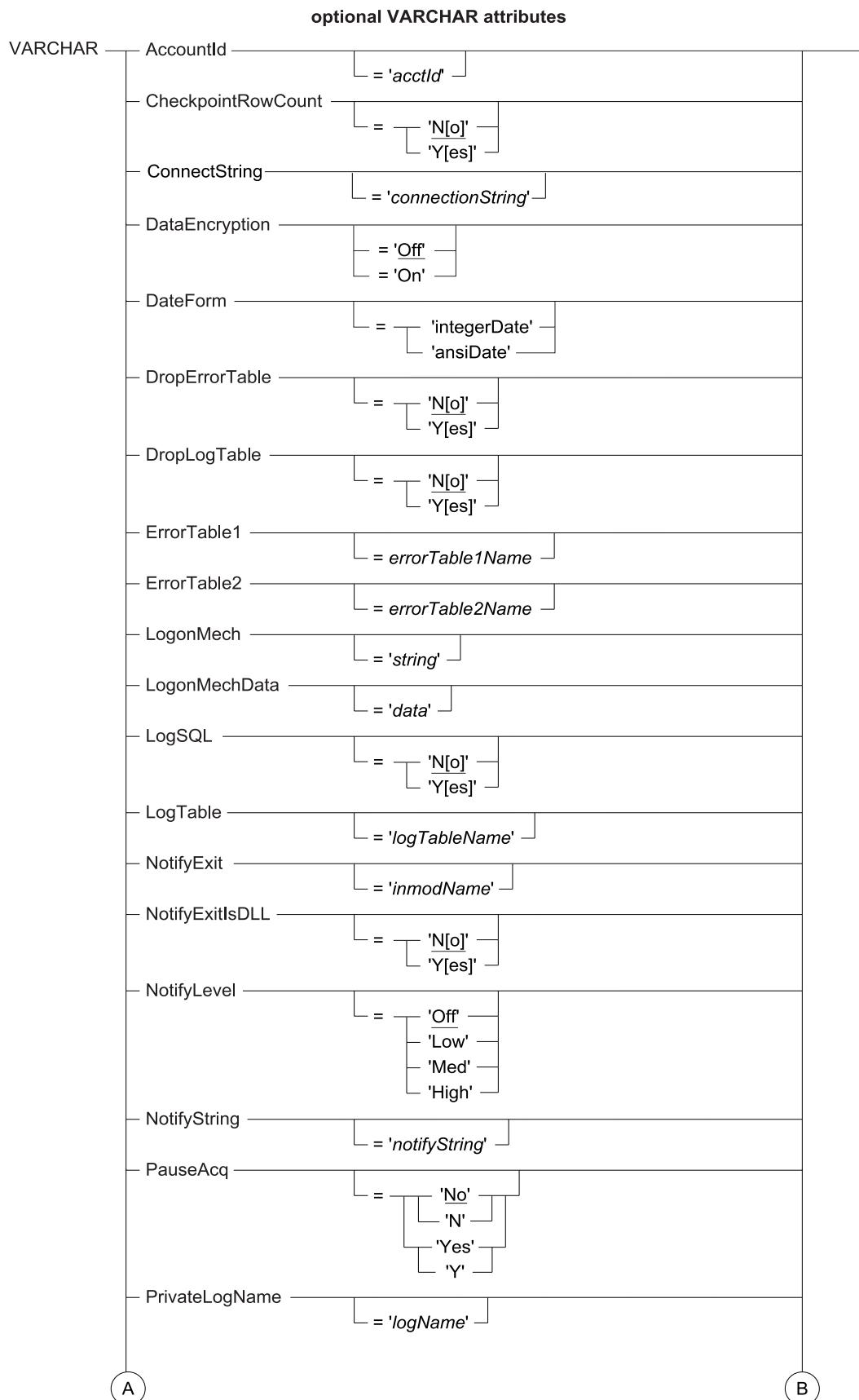
Specification	Description
TYPE	Operator type. Always LOAD or LOAD STANDALONE for the Load operator.
UserName	Operator attribute providing the name of the user for the Load operator logon sessions.
UserPassword	Operator attribute providing the password associated with the user name.
TargetTable	Operator attribute providing the name of the Load target table.

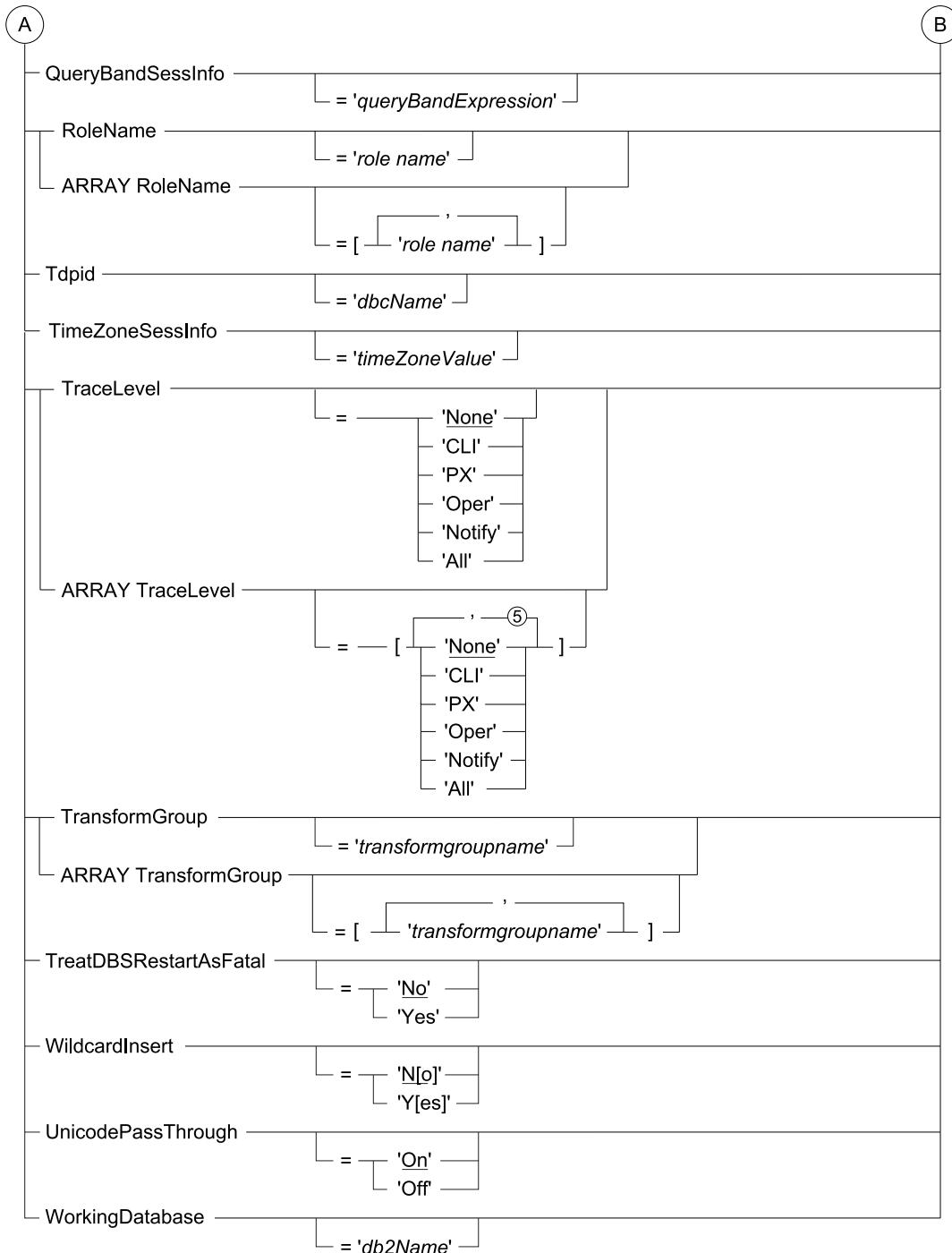
## Required and Optional Attributes

Use the attribute definition list syntax in the Teradata PT DEFINE OPERATOR statement to declare the required and optional attribute values for the Load operator.







**optional VARCHAR attributes (continued)**

where:

### Load Operator Attribute Definitions

Syntax Element	Description
AccountId = 'acctId'	<p>Optional attribute that specifies the account associated with the specified user name.</p> <p>If omitted, it defaults to the account identifier of the immediate owner database.</p>
BufferSize = KBytes	<p>Optional attribute that specifies the output buffer size, in kilobytes, that is used for sending Load parcels to the database.</p> <p>The output buffer size and the size of the rows in the Load table determine the maximum number of rows that can be included in each parcel to the database. A larger buffer size reduces processing overhead by including more data in each parcel.</p> <p>Allowable values are 1 through 16384, but if you specify a value of 16384, the actual buffer size gets set to 16775552 bytes, which is less than the full 16MB. If the value less than 1, an error message results and job terminates.</p> <p>The default buffer size is 1024K bytes when the operator is talking to a Teradata Database 16.00 or later.</p> <p>The default buffer size is 64K bytes when the operator is talking to a pre-16.00 Teradata Database version.</p> <p>The maximum allowed buffer size is usually 16384 Kbytes. Values are evaluated when the connection to the database is made.</p> <p>If the supplied buffer size is too large, the operator scales it back to the maximum allowed buffer size.</p>
CheckpointRowCount - 'option'	<p>Optional attribute that tells the Load operator to enable or disable outputting the rows sent at checkpoints.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>'No' ('N') = The Load operator will not output rows sent at checkpoints (default);</li> <li>'Yes' ('Y') = The Load operator will output rows sent at checkpoints</li> </ul> <p>This attribute is only available in the TPT script mode.</p>
ConnectionString = 'connectionString'	<p>Optional attribute that specifies the connection string. The connection string will be passed to CLI. CLI will validate the connection string.</p> <p>For information on connection string, see <i>Teradata® Call-Level Interface Version 2 Reference for Workstation-Attached Systems</i>, B035-2418.</p> <p><b>Note:</b></p> <p>The TPT Connection String feature is available on all platforms, except for the TDP variant of TPT on z/OS.</p>
DataEncryption = 'option'	<p>Optional attribute that enables full security encryption of SQL requests, responses, and data.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>'On' = all SQL requests, responses, and data are encrypted.</li> <li>'Off' = no encryption occurs (default).</li> </ul>
DateForm = 'option'	<p>Optional attribute that specifies the DATE data type for the Load operator job.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>'integerDate' = the integer DATE data type (default)</li> </ul>

Syntax Element	Description
	<ul style="list-style-type: none"> <li>'ansiDate' = the ANSI fixed-length CHAR(10) DATE data type</li> </ul>
DropErrorTable = ' <i>option</i> '	<p>Optional attribute that specifies whether error tables are dropped upon successful completion of the load job, even if the error tables are empty.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>'Yes' = Load operator will drop the error table only if empty (default). Teradata PT automatically executes a DROP TABLE statement. If you run many load jobs on a regular basis, checking 'Yes' would cause lots of updates to the Data Dictionary, resulting in performance problems.</li> <li>'No' = Load operator will not drop the error table, even if empty. The User must manually execute a DROP TABLE statement on the error table. If the table is not dropped prior to running a Teradata PT job that would use tables by the same name, running that job may result in a database error or in unpredictable results.</li> </ul>
DropLogTable = ' <i>option</i> '	<p>Optional attribute that specifies whether the restart log table is dropped upon successful complete of the load job.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>'Yes' = Load operator will drop the restart log table (default). Teradata PT automatically executes a DROP TABLE statement. You can only set value to 'Yes' for restart jobs. For more information, see the <i>Teradata® Parallel Transporter User Guide</i>, B035-2445.</li> <li>'No' = Load operator will not drop the restart log table. The user must manually execute a DROP TABLE statement on the restart log table. If the table is not dropped prior to running a Teradata PT job that would use the table by the same name, running that job may result in a database error or in unpredictable results.</li> </ul>
ErrorLimit = <i>errorLimit</i>	<p>Optional attribute that specifies the approximate number of records that can be stored in one of the error tables before the Load operator job is terminated.</p> <p>The <i>ErrorLimit</i> specification must be greater than 0. Specifying an invalid value will cause the Load operator to terminate. By default, ErrorLimit value is unlimited.</p> <p>The ErrorLimit specification applies to each instance of the Load operator.</p>
ErrorTable1 = ' <i>errorTable1Name</i> '	<p>Optional attribute that specifies the name of the first error table. ErrorTable1 contains records rejected because of:</p> <ul style="list-style-type: none"> <li>Data conversion errors</li> <li>Constraint violations</li> <li>AMP configuration changes</li> </ul> <p>These types of errors always occur during the acquisition phase of the Load operator job.</p> <p>ErrorTable1 must be a new table name. Do not use a name that duplicates the name of an existing table unless you are restarting a paused Load operator job.</p> <p>The default name for ErrorTable1 is <i>ttname_ET</i>.</p> <p>For more information, see <a href="#">ErrorTable1</a> and <a href="#">Auto-Generation of Error Tables</a>.</p>
ErrorTable2 = ' <i>errorTable2Name</i> '	Optional attribute that specifies the name of the second error table. ErrorTable2 contains records that violated the unique primary index

Syntax Element	Description
	<p>constraint. This type of error always occurs during the application phase of the Load operator job.</p> <p>ErrorTable2 must be a new table name. Do not use a name that duplicates the name of an existing table unless you are restarting a Load operator job. The default name for ErrorTable2 is ttname_UV.</p> <p>For more information, see <a href="#">ErrorTable2</a> and <a href="#">Auto-Generation of Error Tables</a>.</p>
LogonMech = 'string'	<p>Optional attribute that specifies which logon mechanism to use.</p> <p><b>Note:</b></p> <p>Specification of this attribute may be required for some authentication methods.</p> <p>The job terminates if the attribute exceeds 8 bytes.</p> <p>For information on specification requirements for LogonMech, see “Logon Security” in <i>Teradata® Parallel Transporter User Guide</i>, B035-2445.</p>
LogonMechData = 'data'	<p>Optional attribute that passes along additional logon data.</p> <p><b>Note:</b></p> <p>Specification of this attribute is required for some external authentication methods.</p> <p>For information on specification requirements for LogonMechData, see “Logon Security” in <i>Teradata® Parallel Transporter User Guide</i>, B035-2445.</p>
LogSQL = 'option'	<p>Optional attribute that controls how much of the job’s SQL to enter into the log.</p> <p>Valid options are:</p> <ul style="list-style-type: none"> <li>• 'Yes' = output the full SQL to the log. The maximum length is 1M.</li> <li>• 'No' = do not output SQL to the log.</li> <li>• No value or attribute omitted = accept the pre-defined limit, which displays up to 32K of SQL if all the SQL is less than 32K. If the SQL to be logged exceeds 32K, truncate the display to the first 32K bytes.</li> </ul>
LogTable = 'logTableName'	<p>Optional attribute that specifies the name of the restart log table that stores checkpoint information for restarting a job.</p> <p>If running a new job, specify a new table name that is different from the name of any existing table). The Load operator then creates a new restart log table.</p> <p>If restarting a paused job, then the restart log table must exist, and the Load operator restarts the job from the last checkpoint. The restarted job will continue using the existing restart log table.</p> <p>When a job successfully completes, the operator drops the restart log table. Failure to specify a restart log table will cause the job to terminate.</p> <p>The following privileges are required on the restart log table:</p> <ul style="list-style-type: none"> <li>• SELECT</li> <li>• INSERT</li> <li>• DELETE</li> </ul> <p>Users must also have DROP and CREATE privileges on the database that contains the restart log table.</p>

Syntax Element	Description
	<p>The Load operator automatically maintains the restart log table. Manipulating the restart log table in any way invalidates the restart capability.</p> <p>If the restart log table name is not fully qualified, it is created under the user's default (logon) database. If the WorkingDatabase attribute is used, you MUST fully qualify the restart log table name, even if the restart log table is going to reside in the default (logon) database.</p>
MaxSessions = <i>maxSessions</i>	<p>Optional attribute that specifies the maximum number of sessions to log on. The <i>MaxSessions</i> value must be greater than 0. Specifying a value less than 1 terminates the job.</p> <p>The default is one session per available AMP. The maximum value cannot exceed the number of available AMPS.</p>
MinSessions = <i>minSessions</i>	<p>Optional attribute that specifies the minimum number of sessions required for the Load operator job to continue.</p> <p>The <i>MinSessions</i> value must be greater than 0 and less than or equal to the maximum number of Load operator sessions. Specifying a value less than 1 (the default) terminates the job.</p>
NotifyExit = ' <i>inmodName</i> '	<p>Optional attribute that specifies the name of the user-defined notify exit routine with an entry point named <i>_dynamn</i>. If no value is supplied, the following default name is used:</p> <ul style="list-style-type: none"> <li>• libnotifyext.dll for Windows platforms</li> <li>• libnotifyext.dylib for the Apple macOS platform</li> <li>• libnotifyext.so for all other UNIX platforms</li> <li>• NOTIFYEXT for z/OS platforms</li> </ul> <p>See <a href="#">Deprecated Syntax</a> for information about providing your own notify exit routine.</p>
NotifyExitIsDLL = ' <i>option</i> '	<p>Optional attribute (for z/OS systems only) that specifies whether the notify exit routine is built as a DLL (shared library) or not. Valid values are:</p> <ul style="list-style-type: none"> <li>• 'Yes' (or 'Y') = notify exit routine is built as a DLL (default).</li> <li>• 'No' or ('N') = notify exit routine is not built as a DLL.</li> </ul> <p>Any other value terminates the job.</p>
NotifyLevel = ' <i>notifyLevel</i> '	<p>Optional attribute that specifies the level at which certain events are reported. Valid values are:</p> <ul style="list-style-type: none"> <li>• 'Off' = no notification of events is provided (default)</li> <li>• 'Low' = 'Yes' in the Low Notification Level column</li> <li>• 'Med' = 'Yes' in the Medium Notification Level column</li> <li>• 'High' = 'Yes' in the High Notification Level column</li> </ul>
NotifyMethod = ' <i>notifyMethod</i> '	<p>Optional attribute that specifies the method for reporting events. The methods are:</p> <ul style="list-style-type: none"> <li>• 'None' = no event logging is done (default).</li> <li>• 'Msg' = sends the events to a log.</li> </ul> <p>On Windows, the events are sent to the EventLog that can be viewed using the Event Viewer. The messages are sent to the application log.</p> <p>On Solaris, AIX, and Linux platforms, the destination of the events is dependent upon the setting specified in the /etc/syslog.conf file.</p>

Syntax Element	Description
	<p>On SLES11, the destination of the events is dependent upon the setting specified in the /etc/syslog-<i>ng</i>.conf file.</p> <p>On z/OS systems, events are sent to the job log.</p> <ul style="list-style-type: none"> <li>'Exit' = sends the events to a user-defined notify exit routine. Row count information is in 4-byte unsigned integer values.</li> <li>'Exit64' = sends the events to a user-defined notify exit routine. Row count information is in 8-byte unsigned integer values for these events: NFEEventCheckPoint64 NFEEventPhase1End64 NFEEventPhase1End64 NFEEventDropErrTable164 NFEEventDropErrTable1I64</li> <li>'ExitEON' = sends the events to a user-defined notify exit routine. Complete Teradata object names are passed to the notify exit routine for these events: NFEEventInitializeEON NFEEventPhase1BeginEON In addition, ExitEON sends row count information in 8-byte unsigned integer values.</li> </ul>
NotifyString = ' <i>notifyString</i> '	<p>Optional attribute that specifies a user-defined string to precede all messages sent to the system log. This string is also sent to the user-defined notify exit routine. The maximum length of the string is:</p> <ul style="list-style-type: none"> <li>80 bytes, if NotifyMethod is 'Exit'</li> <li>16 bytes, if NotifyMethod is 'Msg'</li> </ul>
PauseAcq = ' <i>option</i> '	<p>Optional attribute that specifies whether to pause the Load operator job after the acquisition phase or enter the application phase. Values are:</p> <ul style="list-style-type: none"> <li>'N[o]' for normal Load operator jobs, to distribute all the rows that were sent to the database during the acquisition phase to their final destination on the AMPs (default).</li> <li>'Y[es]' to pause after the completion of the acquisition phase and skip the application phase.</li> </ul> <p>Specifying any other value terminates the job.</p> <p>The absence of any value for the PauseAcq attribute means that the Load operator job will execute both the acquisition phase and the application phase without pausing. This will distribute all the rows that were sent to the database during the acquisition phase to their final destination on the AMPs.</p>
PrivateLogName = ' <i>logName</i> '	<p>Optional attribute that specifies the name of a log that is maintained by the Teradata PT Logger inside the public log. The private log contains all the output provided by the operator.</p> <p>The private log can be viewed using the tlogview command as follows, where <i>jobId</i> is the Teradata PT job name and <i>privateLogName</i> is the value for the operator's PrivateLogName attribute:</p> <pre>tlogview -j <i>jobid</i> -f <i>privatelogname</i></pre> <p>If the private log is not specified, all output is stored in the public log.</p>

Syntax Element	Description
	<p>By default, no diagnostic trace messages are produced. Diagnostic trace messages are produced only when the user sets a valid value for the TraceLevel attribute.</p> <p>For more information about the tlogview command, see <a href="#">Teradata PT Utility Commands</a>.</p>
QueryBandSessInfo = 'queryBandExpression'	<p>Optional attribute that specifies the Query Band for the duration of the job sessions.</p> <p>The <i>queryBandExpression</i> is a set of name=value pairs, separated by a semicolon and ending with a semicolon. The user defines the Query Band expression, which will look similar to the following example:</p> <pre>'org=Finance;load=daily;location=west;'</pre> <p>QueryBandSessInfo may also be specified as an ARRAY attribute.</p> <p>For information on the rules for creating a Query Band expression, see <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i>, B035-1144 and <i>Teradata Vantage™ - SQL Data Definition Language Detailed Topics</i>, B035-1184.</p> <p>The value of the QueryBandSessInfo attribute is displayed in the Load operator private log.</p> <p>Use of the QueryBandSessInfo attribute is subject to the following rules:</p> <ul style="list-style-type: none"> <li>• By default, Query Band is off until a valid value appears for the QueryBandSessInfo attribute.</li> <li>• If the QueryBandSessInfo attribute contains a value, the Load operator constructs the necessary SET QUERY BAND SQL and issues it as part of the Load operator SQL sessions to communicate the request to the database.</li> <li>• The Load Operator does not check the Query Band expression, but passes the expression to the database as is.</li> <li>• If the version of the database against which the job is being run does not support the Query Band feature, no Query Banding will take place. However, the operator will ignore the error and run the rest of the job.</li> <li>• If there is a syntax error in the Query Band expression, the database will return an error. The Load operator will then terminate the job and report the error to the user.</li> </ul>
RoleName = 'role name'	<p>Optional attribute that implements security in a database environment. The operator will prepend the value with "SET ROLE ". The syntax will be sent to the database as follows:</p> <pre>SET ROLE &lt;role name&gt;;</pre> <p>For example:</p> <pre>SET ROLE All;</pre> <p>For details of "SET ROLE" command use, see <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i>, B035-1144.</p> <p>The VARCHAR ARRAY can specify more than one value, for example:</p> <pre>VARCHAR ARRAY RoleName = [ 'role name1', 'role name2' ],</pre> <p>The operator will send the request to the database on the main control session and the auxiliary SQL session after the sessions are connected.</p>

Syntax Element	Description
	<p>The operator does not send the request on the FastLoad protocol sessions, because the database does not allow the request to be send on the FastLoad protocol sessions.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>The operator does not validate the value for this attribute. The operator passes the value to the database as is. The database will validate the value. The operator will terminate the job with an error when the validation fails.</li> <li>C-style comments are allowed in the value and will be passed to the database.</li> <li>ANSI-style comments are not supported in the value. The operator can terminate the job with a syntax error when the value contains an ANSI-style comment.</li> <li>A semicolon is not allowed in the value, because the operator allows only a single statement per request. The operator will terminate the job with an operator error when the value contains a semicolon.</li> </ul>
TargetTable = ' <i>targetTableName</i> '	<p>Required attribute that specifies the name of the Load target table to receive the data from the client system.</p> <p>A target table must already exist on a database and be empty, with no defined secondary or join indexes, before a Load job is run. You cannot use a name that duplicates the name of an existing table unless you are restarting a paused Load operator job.</p> <p>For target tables defined as NoPI, the restriction that the target table must be empty does not apply.</p>
VARCHAR TASMFASTFAIL = ' <i>value</i> '	<p>Optional attribute that enables FASTFAIL feature.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>'Yes' or 'Y' = Enable FastFail feature. The job will terminate gracefully when it is supposed to be held by the database.</li> <li>'No' or 'N' = FastFail feature is not enabled (default). The job will appear to hang if the TASM rules dictate that a job should be held for any reason.</li> </ul>
TenacityHours = <i>hours</i>	<p>Optional attribute that specifies the number of hours that the Load operator continues trying to log on when the maximum number of load/unload operations are already running on the database.</p> <p>The default value is 4 hours. To enable the tenacity feature, <i>hours</i> must be greater than 0. Specifying a value of 0 disables the tenacity feature. Specifying a value of less than 0 terminates the Load job.</p>
TenacitySleep = <i>minutes</i>	<p>Optional attribute that specifies the number of minutes that the Load operator pauses before retrying to log on when the maximum number of load/export operations are already running on the database.</p> <p>The minutes value must be greater than 0. If you specify a value less than 1, the Load operator responds with an error message and terminates the job. The default is 6 minutes.</p>
Tdpld = ' <i>dbcName</i> '	Optional attribute that specifies the name of the database machine (non-mainframe platforms) or TDP (mainframe platforms) for the Load operator job.

Syntax Element	Description
	<p>The <code>dbcName</code> can be up to 256 characters and can be a domain server name.</p> <p>If you do not specify the value for the <code>Tdpld</code> attribute, the operator uses the default <code>Tdpld</code> established for the user by the system administrator.</p>
<code>TimeZoneSessInfo = 'timeZoneValue'</code>	<p>Optional attribute that allows you to change the default time zone displacement for the duration of the operator's job session.</p> <p>When you provide a value for this attribute, the operator will build the <code>SET TIME ZONE &lt;timeZoneValue&gt;;</code> SQL request.</p> <p>The operator will send the request to the database on the main control and auxiliary SQL sessions after the sessions are connected.</p> <p>The operator does not send the request on the FastLoad protocol sessions, because the database does not allow the request to be sent on the FastLoad protocol sessions.</p> <p>Here are some examples:</p> <ul style="list-style-type: none"> <li><b>Example 1:</b> This example sets the session default time zone displacement to <code>LOCAL</code>, which is the system default time zone:  <code>VARCHAR TimeZoneSessInfo = 'LOCAL'</code></li> <li><b>Example 2:</b> This example sets the session default time zone displacement to <code>USER</code>, which is the default time zone for the logged on user:  <code>VARCHAR TimeZoneSessInfo = 'USER'</code></li> <li><b>Example 3:</b> This example sets the session default time zone displacement to a simple constant time zone string expression:  <code>VARCHAR TimeZoneSessInfo = '''America Pacific'''</code></li> </ul> <p><b>Note:</b></p> <p>Any single quote character ('') inside the value must be entered as two consecutive single quote characters in a TPT job script. This ensures the correct value will be sent to the database.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>The operator does not validate the value for this attribute. The operator passes the value to the database as is. The database will validate the value. The operator will terminate the job with an error when the validation fails.</li> <li>C-style comments are allowed in the value and will be passed to the database.</li> <li>ANSI-style comments are not supported in the value. The operator will terminate the job with a syntax error when the value contains an ANSI-style comment.</li> <li>A semicolon is not allowed in the value, because the operator allows only a single statement in the "SET TIME ZONE SQL" request. The operator will terminate the job with an operator error when the value contains a semicolon.</li> </ul> <p>For more information on SET TIME ZONE SQL, see <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i>, B035-1144.</p>
<code>TraceLevel = 'level'</code>	Optional attribute that specifies the types of diagnostic messages that are written by each instance of the operator to the public log (or private log, if one is specified using the <code>PrivateLogName</code> attribute). The diagnostic trace

Syntax Element	Description
	<p>function provides more detailed information in the log file to aid in problem tracking and diagnosis. The trace levels are:</p> <ul style="list-style-type: none"> <li>'None' = TraceLevel turned off (default).</li> <li>'CLI' = enables the tracing function for CLI-related activities (interaction with the database)</li> <li>'PX' = enables the tracing function for activities related to the Teradata PT infrastructure</li> <li>'Oper' = enables the tracing function for operator-specific activities</li> <li>'Notify' = enables the tracing of activities related to the Notify feature</li> <li>'All' = enables tracing for all these activities</li> </ul> <p>The VARCHAR ARRAY can specify more than one value, for example:</p> <pre>VARCHAR TraceLevel = 'CLI' VARCHAR TraceLevel = 'OPER' VARCHAR ARRAY TraceLevel = [ 'CLI' ] VARCHAR ARRAY TraceLevel = [ 'CLI', 'OPER' ]</pre> <p><b>Note:</b></p> <p>The TraceLevel attribute is provided as a diagnostic aid only. The amount and type of additional information provided by this attribute changes to meet evolving needs from release to release.</p>
TransformGroup = ' <i>transformgroupname</i> '	<p>Optional attribute that supports changing the active transform for Teradata Complex Data Types (CDTs). The value is the <i>&lt;udt name&gt; &lt;transform group name&gt;</i>, and the operator will prepend the hard-coded string "SET TRANSFORM GROUP FOR TYPE ". The syntax sent to RDBMS is as follows:</p> <pre>SET TRANSFORM GROUP FOR TYPE &lt;udt name&gt; &lt;transform group name&gt;;</pre> <p>For example:</p> <pre>"SET TRANSFORM GROUP FOR TYPE JSON CHARACTER SET LATIN TD_JSON_VARCHAR;"</pre> <p>The VARCHAR ARRAY can specify more than one value, for example:</p> <pre>VARCHAR ARRAY TransformGroup = [ 'JSON CHARACTER SET LATIN TD_JSON_VARCHAR',                                 'ST_GEOGRAPHY TD_GEO_VARCHAR' ],</pre> <p>The operator will send the request to the database on the main control session after the session is connected.</p> <p>The operator does not send the request on the auxiliary SQL session, because the database does not require it.</p> <p>The operator does not send the request on the FastLoad protocol sessions, because the database does not allow the request to be send on the FastLoad protocol sessions.</p>

Syntax Element	Description
	<p><b>Note:</b></p> <ul style="list-style-type: none"> <li>The operator does not validate the value for this attribute. The operator passes the value to the database as is. The database will validate the value. The operator will terminate the job with an error when the validation fails.</li> <li>C-style comments are allowed in the value and will be passed to the database.</li> <li>ANSI-style comments are not supported in the value. The operator can terminate the job with a syntax error when the value contains an ANSI-style comment.</li> <li>A semicolon is not allowed in the value, because the operator allows only a single statement per request. The operator will terminate the job with an operator error when the value contains a semicolon.</li> </ul>
TreatDBSRestartAsFatal= ' <i>option</i> '	<p>Optional attribute that tells the operator whether to terminate the job when a database restart occurs.</p> <p>The TreatDBSRestartAsFatal values are:</p> <ul style="list-style-type: none"> <li>'No' ('N') = The operator will not terminate if a database restart occurs (default). The database restart will be treated as a retryable one.</li> <li>'Yes' ('Y') = The operator will terminate if a database restart occurs.</li> </ul>
UnicodePassThrough = ' <i>value</i> '	<p>Optional attribute that tells the operator to enable or disable the Unicode Pass Through feature.</p> <p>Valid values:</p> <ul style="list-style-type: none"> <li>'On' = Enable the Unicode Pass Through feature in the operator.</li> <li>'Off' = (Default) Disable the Unicode Pass Through feature in the operator.</li> </ul> <p><b>Note:</b></p> <p>When a TPT job is using the UTF8 or UTF16 session character set, the UnicodePassThrough attribute can be set to 'On' to allow the operator to load data with Unicode pass through characters.</p>
UserName = ' <i>userId</i> '	<p>Attribute that specifies the database user name.</p> <p><b>Note:</b></p> <p>Use of this attribute is not compatible with some external authentication logon methods. For more information on UserName specification requirements, see "Logon Security" in <i>Teradata® Parallel Transporter User Guide</i>, B035-2445.</p>
UserPassword = ' <i>password</i> '	<p>Attribute that specifies the password associated with the user name.</p> <p><b>Note:</b></p> <p>Use of this attribute is not compatible with some external authentication logon methods. For more information on password specification requirements, see "Logon Security" in <i>Teradata® Parallel Transporter User Guide</i>, B035-2445.</p>
WildcardInsert= ' <i>option</i> '	Optional attribute that builds an INSERT statement from the table definition.

Syntax Element	Description
	<p>Use this attribute to load all the columns in a table when the columns contain user-defined types (UDTs).</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>‘Y[es]’ = builds an INSERT statement.</li> <li>‘N[o]’ = no activity (default).</li> </ul> <p>If you set this attribute to Yes when a valid, fully supported INSERT statement already exists, an error results.</p> <p>A valid tableName matches the name of the table used in the TargetTable attribute and the semicolon is in the last non-whitespace character in the supplied DML statement as follows:</p> <pre>INS[ERT] [INTO] &lt;tablename&gt; ;</pre>
WorkingDatabase = 'databaseName'	<p>Optional attribute that specifies a database other than the logon database as the default database.</p> <p>The name of the database that is specified with this attribute is used in the SQL DATABASE statement that is sent by the operator immediately after connecting the two SQL sessions.</p> <p>If WorkingDatabase is not specified, the default database associated with the logged on user is assumed for all unqualified table names.</p>

## Usage Notes

### LogTable

A restart log table, which contains restart information written during job runs, is required for any execution of the Load operator. Specify a restart log table in scripts with the LogTable attribute.

Restarts (as discussed in [Staged Loading](#)) are common during staged loading operations. When additional data is available after a job is paused, the job is restarted by submitting a second script that specifies the additional data. The Load operator recognizes the job as a restart, reads the restart log to determine the status of the job, then loads the additional file.

Restarts can also occur following any unexpected error on a database. For example, if a table runs out of available space during a load operation, the job terminates, the table is paused, and a checkpoint is recorded in the restart log. Pausing the job in this way allows you to manually increase the available space, if needed, then restart the job because the load operation can restart a job from the last checkpoint in the restart log.

If you do not specify a name for *LogTable*, the Load operator automatically creates a name of the log table as follow:

```
ttname_RL
```

where *ttname* is the name of the corresponding target table.

**Note:**

The value of the TargetTable attribute is truncated to the maximum number of characters for object names that the database supports, minus 3 characters before the suffix "\_RL" is appended to the target table name. This means that if the value of the TargetTable attribute is a fully qualified table name and that fully qualified name exceeds the maximum supported length of a database object, the generated name for the log table may not be what you intend. In such a case, Teradata recommends that you provide the names of the log table and not rely on the Load operator to generate the names for log table automatically.

## ErrorTable1

Load operations create an error table that captures errors during job runs. Jobs can use the default names of the error tables, or names can be user-specified as an attribute of the Load operator.

Error Table1 contains most of the errors connected with data and the data environment. The following types of errors are captured:

- **Constraint violations** – Records that violate a range or value constraint defined for specific columns of a table.
- **Unavailable AMP** – Records to be written to a non-fallback table about an offline AMP.
- **Data conversion errors** – Records that fail to convert to a specified data type.

**Note:**

The names in the ErrorTable1 ErrorFieldName column have a maximum supported size of 120 characters. The names can be up to 128 characters, but if a row is inserted into ErrorTable1, the database truncates any name that exceeds 120 characters.

## ErrorTable2

Load operations create a duplication error table that captures errors during job runs. Jobs can use the default names of the error tables, or names can be user-specified as an attribute of the Load operator.

If you specify a name for ErrorTable2, the system creates the table in the default database for the logon user, unless a different database is specified in the WorkingDatabase attribute.

If you do not specify a name for ErrorTable2, the system default name for ErrorTable2 is based on the TargetTable name. The system names the table using the form tname\_UV and creates the table in the database that contains the TargetTable, unless a different database is specified in the WorkingDatabase attribute. This error table is not used when the target table has a non-unique primary index.

## Auto-Generation of Error Tables

Some tables must be created by the user before the job begins and some are created during the execution of a job. A target table must exist on the database when the Load operator job is executed.

If you have not specified error tables (specifying them is optional), the Load operator automatically creates names of the error tables as follows:

- The first error table is *ttnname*\_ET
- The second error table is *ttnname*\_UV

where *ttnname* is the name of the corresponding target table.

**Note:**

The value of the TargetTable attribute is truncated to the maximum number of characters for object names that the database supports, minus 3 characters before the suffixes "\_ET" and "\_UV" are appended to target table names. This means that if the value of the TargetTable attribute is a fully qualified table name and that fully qualified name exceeds the maximum supported length of a database object, the generated names for the error tables may not be what you intend. In such a situation, Teradata recommends that you provide the names of the error tables and not rely on the Load operator to generate the names for these tables automatically.

For example, if the following is specified when no error table specifications exist,

```
VARCHAR TargetTable = 'ttnname'
```

the Load operator creates the following error tables:

```
targtable_ET  
targtable_UV
```

## Normalized Tables

The Load operator does not support normalized tables.

A normalized table is a table that has been created using the NORMALIZE clause in the CREATE TABLE statement. To normalize means to combine two period values that meet or overlap.

## Restrictions and Limitations

### Table Characteristics and Considerations

The target table cannot have any of the following characteristics:

- Join, hash, or secondary indexes
- Foreign key references
- Referential integrity
- Triggers

- Cannot be a normalized table
- Cannot be a column partitioned table

## Data Types

- The job cannot load the following data types:
  - LOB
  - JSON
  - XML
  - ST\_Geometry
- UDT data can be loaded in its external type format.

## Error Tables

- Error tables cannot exist for new jobs.
- The table names for the ErrorTable1 and ErrorTable2 attributes must be new tables unless you are restarting a paused Load operator job.

## Other Notes

- One Load operator job can load a single database table.
- The target table must already exist and be empty for new jobs. The empty table restriction applies to PI and NoPI tables.
- The target table is locked until the application phase is complete.
- The target table cannot be rolled back after the application phase is complete.
- Duplicate rows are discarded by the database for SET and MULTISET tables. The exception is NoPI tables. Duplicate rows are not discarded for NoPI tables.
- Only the SQL INSERT statement is supported. Any other SQL statements are not supported.
- The job cannot load data into a view.
- The Load operator requires one database load slot.

## Job Options

### Duplicate Rows

Duplicate rows, which are exact duplicates of existing table rows, are never inserted, even if the target table is defined as a multiset table, which usually permits duplicate rows. Duplicate row violations are thus not captured in either Error Table 1 or Error Table 2. Instead, they are counted and reported in the status log at the end of a job.

If the target table is defined as a NoPI table, duplicate rows are inserted. NoPI tables are inherently multiset since no duplicate row checking is possible; with NoPI tables, duplicate rows can be on different AMPs. Therefore, no duplicate row elimination is performed.

If a table has a unique primary index, a circumstance where there is duplicate row takes precedence over a duplicate primary index error, meaning that the offending row is counted and reported, but it is not captured in Error Table 2.

## ErrorLimit

While loading a large amount of data, a single data error might be repeated for each input record. Because an error can often be corrected long before errors are generated for all the records in a job run, consider using the ErrorLimit attribute to specify a limit to the number of errors that can be tolerated before a job is terminated.

This limit, which is specified with the ErrorLimit attribute, represents the total number of errors written to the first error table per instance of the Load operator, not to all instances combined. Therefore, if an error limit is set to 1,000, a single load instance must detect that 1,000 rows are inserted into the error table before the job is terminated.

The error limit can also be reached at checkpoint time.

## Error Limit Examples

To illustrate how the Load operator determines if the number of errors has reached the Error Limit, consider these examples if there are two instances running and the Error Limit has been set to 1000.

- If either instance by itself reaches 1000, it will terminate the job by returning a fatal error.
- If instance #1 processes 500 error rows and instance #2 processes 500 error rows but does not reach a checkpoint. The job will continue processing.
- If instance #1 processes 500 error rows and instance #2 processes 500 error rows but does reach a checkpoint. The total number of error rows for all instances combined is determined at checkpoint time and at the end of the Acquisition Phase. If the total of all instances exceeds the error limit at that time, the job will terminate with an error.

## Staged Loading

Staged loading is the ability to pause an active load operation until additional data is available.

If a single table needs to be filled with the contents of three files, for example, usually all three files are streamed to look like a single source to Teradata PT. But if one of the files will not exist until the next day, it is possible to load in stages. In other words, Teradata PT can load the first two files, pause the Load operation, and then load the third file when it is available.

Staged loading is set by the attribute `PauseAcq = 'Y'`, which prevents the Load operator from proceeding to the application phase. Each stage of the load operation is accomplished with a separate job script: one for the acquisition phase, and one for the application phase.

For example, to accomplish the scenario with three files (one of which is unavailable until the next day), run Job1 on Day 1 using the two existing files as input to the load, with the `PauseAcq = 'Y'` setting.

When this stage of the job is finished, the target table is paused and becomes inaccessible to users. Attempts to access the target table (or the error tables) return the following error message:

Operation Not Allowed <tablename> is being loaded

On Day 2, restart the paused job by running Job2 using the third, newly available file as input. For this stage, set `PauseAcq = 'N'`. When this stage of the job finishes, the table is fully loaded and ready for access.

### NOTICE

A paused table, though inaccessible, can be dropped.

## Operational Considerations

### Performance

Multiple parallel instances can be used to improve the performance of the load. Multiple instances of the Load operator can be run in a job, but all of them load data into the same table.

During a Load job, the Load operator inserts the data from each record it receives from the data streams into one row of the database target table. Because Teradata PT can filter out rows based on criteria in the script, some input rows may not be loaded.

The Load operator does not support Update, Select, and Delete operations on the target table. The Load operator only support Insert operations.

### Space Requirements

Always estimate the final size of the Load target table to ensure that the destination on a database has enough space to accommodate a Load operator job.

If the system that owns a Load target table, log table, or error tables runs out of space, the database returns an error message, then the Load operator job terminates. Additional space must be allocated to the database before the job can be restarted.

### Sessions and Instances

The Load operator will connect one main (control) SQL session and one auxiliary SQL session, in addition to the data sessions. The main SQL session is responsible for executing SQL statements

pertaining to utility work. The auxiliary session is used for creating and maintaining a restart log table for recovery purposes.

A minimum and a maximum number of sessions (the session limits) can be specified for the Load operator.

Consider the following usage notes:

- The maximum sessions connected can never exceed the number of available AMPs in the system, even if a larger number is specified.
- The default is one session per available AMP.
- For the MinSessions attribute, the minimum specification is one.
- The MaxSessions attribute can be set to a number smaller than the number of AMPs on the database server if fewer sessions are suitable for the job.
- Network protocol software might also impose limits on workstation-attached systems.
- Platform limits for maximum sessions per application differ:
  - On a mainframe-attached z/OS client system, use the TDP SET MAXSESSIONS command to specify a platform limit.
  - On workstation-attached client systems for UNIX, Linux, and Windows systems, this value is defined in the CLI file, clispb.dat, under the max\_num\_sess variable.
  - On mainframe-attached z/OS client systems, this value is defined in the HSHSPB parameter under the IBCSMAX setting.

The *max\_num\_sess* value in the clispb.dat file (or HSHSPB) specifies the total number of sessions allowed to be connected by a single application at one time. The *max\_num\_sess* pertains to all sessions connected, both SQL and data loading.

## Limits on Load Jobs

The Load operator requires one database load slot.

The number of concurrent load slots is, however, configurable in the database environment using the same MaxLoadTasks field control used by FastLoad, FastExport, and MultiLoad. For example, one Teradata PT Load job equates to one FastLoad job in the count for MaxLoadTasks. For more information, see “DBS Control Utilities” in *Teradata Vantage™ - Database Utilities*, B035-1102.

 **NOTICE**

Simultaneously running many Teradata PT jobs can impact other running database processes and applications.

## Checkpointing and Restarting

The Load operator takes checkpoints at the beginning and end of the acquisition phase. More granular checkpoints during the acquisition phase can be specified using the command line option -z when running Teradata PT using the **tbuild** command. The **-z** option specifies checkpoint intervals in terms of seconds.

The following command string is an example of the -z option:

```
tbuild -f <file name> -z 30
```

In this command, the **-f** option indicates the script that is input to **tbuild**, and the **-z** option indicates that a checkpoint will be taken every 30 seconds.

The **DEFINE JOB** statement can also be used to specify a checkpoint value.

Checkpointing during the application phase is managed internally by Analytics Database, and therefore is *not* user-controlled.

## Load Operator as Standalone Operator

In addition to its other capabilities, the Load operator can function as a standalone operator and supports an **APPLY** statement with no **SELECT** statement and no **INSERT** statement. To use **LOAD** as a standalone operator, use one of the following definitions:

```
TYPE LOAD STANDALONE
TYPE LOAD
```

---

### Note:

The **STANDALONE** keyword is optional.

---

In the following example, the **STANDALONE** keyword is omitted:

```
DEFINE JOB LOAD_USER_DATA
(
    DEFINE OPERATOR LOAD_OPERATOR
    TYPE LOAD
    (
        .
        .
        .
    );
    .
    .
    .
    APPLY
    TO OPERATOR (LOAD_OPERATOR[1]);
);
```

---

**Note:**

Use LOAD as a standalone operator to apply data on the target table without sending more data.

---

# MultiLoad INMOD Adapter Operator

## MultiLoad INMOD Adapter Operator Capabilities

The MultiLoad INMOD Adapter operator allows you to use Teradata MultiLoad utility INMOD routines with Teradata PT. This operator can be used as a producer or a filter operator.

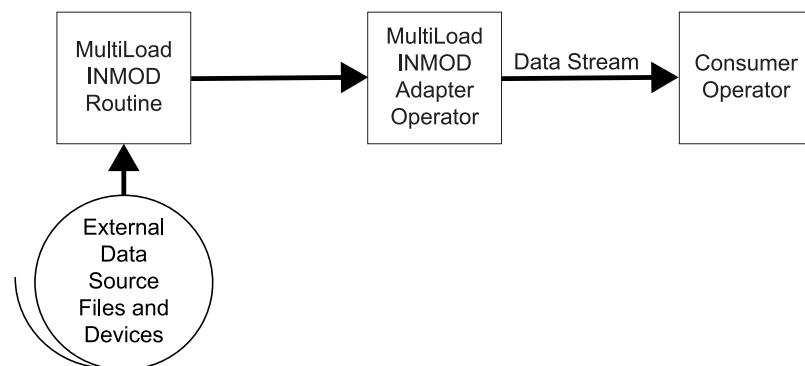
The term INMOD is an acronym for “*input modification*.” INMOD routines are user exit routines the MultiLoad INMOD adapter can call to provide enhanced processing functions on input records before they are sent to the database.

You can use the MultiLoad INMOD Routine with the MultiLoad INMOD Adapter to read and preprocess input data values from files on the client system.

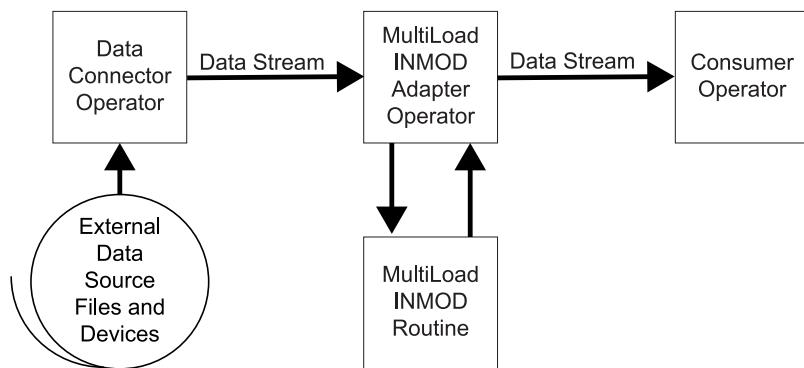
For information on creating and using MultiLoad INMOD Routines, see *Teradata® MultiLoad Reference*, B035-2409.

The following two figures show the MultiLoad INMOD Adapter operator as a producer operator or as a filter operator.

### ***MultiLoad INMOD Adapter Operator as Producer Operator***



### ***MultiLoad INMOD Adapter Operator as Filter Operator***



#### **Note:**

- To use the MultiLoad INMOD Adapter Operator as a filter operator, define the operator as TYPE MULTILOAD INMOD FILTER in the DEFINE OPERATOR statement and then use the VIA clause to invoke the operator in the APPLY statement. For VIA clause usage, see [Object Definitions and the APPLY Statement](#).
- The existing TPT function names should not be used in User's INMODs. Using the existing TPT function names in User's INMODs might cause the normal operation of the TPT job to behave with unexpected results and prevent requests from completing normally.

## Syntax

Use the following specifications and attributes to define the MultiLoad INMOD Adapter operator in a Teradata PT job script.

## Required Specifications

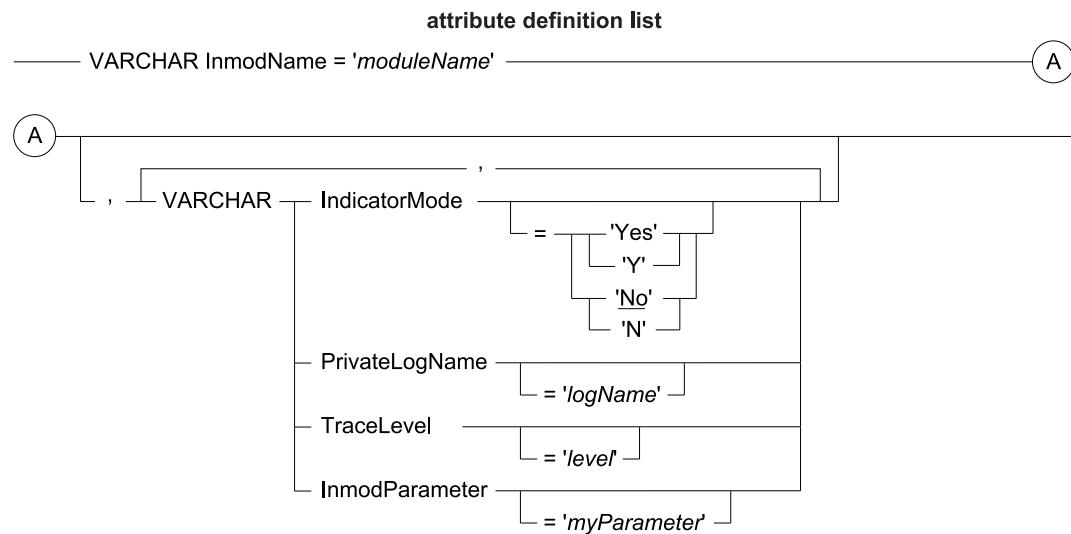
The following specifications are required to define a MultiLoad INMOD Adapter job.

#### **Required Syntax for the MultiLoad INMOD Adapter Operator**

Specification	Description
TYPE	Operator type. MULTILOAD INMOD for the MultiLoad INMOD Adapter operator as a producer operator; MULTILOAD INMOD FILTER for the MultiLoad INMOD Adapter operator as a filter operator.
InmodName	Operator attribute providing the name of the INMOD routine.

## Required and Optional Attributes

Use the attribute definition list syntax in the Teradata PT DEFINE OPERATOR statement to declare the required and optional attribute values for the MultiLoad INMOD Adapter operator.



where:

### MultiLoad INMOD Adapter Operator Attribute Descriptions

Syntax Element	Description
IndicatorMode = 'option'	<p>Optional attribute that specifies whether indicator byte(s) is included at the beginning of each record (when the MultiLoad INMOD Adapter operator is used as a producer operator).</p> <ul style="list-style-type: none"> <li>'Yes' (or 'Y') = indicator mode data</li> <li>'No' (or 'N') = nonindicator mode data (default)</li> </ul> <p>When the MultiLoad INMOD Adapter operator is used as a filter-type operator, this is the optional attribute that specifies whether the MultiLoad routine can handle indicator byte(s).</p> <ul style="list-style-type: none"> <li>'Yes' (or 'Y') = can handle</li> <li>'No' (or 'N') = cannot handle (default)</li> </ul>
InmodName = 'moduleName'	<p>Required attribute that specifies the name of the MultiLoad INMOD routine.</p> <p>When you specify the name in a Teradata PT job script, it is the named routine rather than a data source that provides the input data records.</p> <p>The INMOD is found in the directory defined in the system-dependent library path. Using the "./" syntax (for example ./mlinmod) indicates the directory in which the job is run.</p>
InmodParameter = 'myParameter'	Optional attribute that specifies the second parameter of the MultiLoad INMOD routines.

Syntax Element	Description
PrivateLogName = 'logName'	<p>Optional attribute that specifies the name of a log that is maintained by the Teradata PT Logger inside the public log. The private log contains all the diagnostic trace messages produced by the operator.</p> <p>The file name is appended with the operator instance number. A “-1” is appended to the log name for instance 1. For example, if PrivateLogName = 'MLINlog', then the actual log name for instance 1 is MLINlog-1. Similarly, for instance 2, it is MLINlog-2, and so on.</p> <p>The private log can be viewed using the tlogview command as follows, where jobId is the Teradata PT job name and privateLogName is the value for the operator's PrivateLogName attribute:</p> <pre>tlogview -j jobId -f privateLogName</pre> <p>If the private log is not specified, all output is stored in the public log.</p> <p>For more information about the tlogview command, see <a href="#">Teradata PT Utility Commands</a>.</p>
TraceLevel = 'level'	<p>Optional attribute that specifies the types of diagnostic messages that are written by each instance of the operator to the public log (or private log, if one is specified using the PrivateLogName attribute). The diagnostic trace function provides more detailed information in the log file to aid in problem tracking and diagnosis.</p> <p>The trace levels are:</p> <ul style="list-style-type: none"> <li>• 'None' = Enables minimal information such as initialization, access module attach /detach operations, file opening, error messages, and statistics (number of records processed and processor time used). This is the default.</li> <li>• 'Milestones' = enables the trace function only for major events such as initialization, file openings and closings, error conditions, and so on.</li> <li>• 'IO_Counts' = enables the trace function for major events plus I/O counts</li> <li>• 'IO_Buffers' = enables the trace function for major events and I/O counts plus I/O buffers</li> <li>• 'All' = enables the trace function for major events and I/O counts and buffers plus function entries</li> </ul> <p>If you use the PrivateLogFile attribute to specify a log file but do not specify the TraceLevel attribute, the following minimal statistics are displayed:</p> <ul style="list-style-type: none"> <li>• Name of files as they are processed</li> <li>• Notice when sending rows begins</li> <li>• Upon completion of each file, the number of rows processed and the CPU time consumed.</li> <li>• Total files processed and CPU time consumed by each instance of the DataConnector operator.</li> </ul> <p><b>Note:</b></p> <p>The TraceLevel attribute is provided as a diagnostic aid only. The amount and type of additional information provided by this attribute will change to meet evolving needs from release to release.</p> <p>When the MultiLoad INMOD Adapter operator is used as a filter-type operator, it does not show any diagnostic messages.</p>
VARCHAR	Keyword that specifies VARCHAR as the data type of the defined attribute.

## Usage Notes

Consider the following information when defining a MultiLoad INMOD Adapter operator.

### Input Record Format

The internal format of each input record intended for use with the MultiLoad INMOD Adapter operator must conform to the specified schema.

The MultiLoad INMOD Adapter operator requires each field to be of a specified length. When no length is specified, the operator assumes the conventional default length.

The MultiLoad INMOD Adapter operator supports both fixed- and variable-length input records.

For records with a two-column schema such as:

```
(  
  COL1 INTEGER,  
  COL2 VARCHAR(10));
```

each record comprises:

- A four-byte integer, because 4 is the INTEGER length,
- A two-byte length indicator for the VARCHAR field, and
- The CHAR data of the length indicated in the two-byte length indicator field.

The following examples show the byte order when running Teradata PT on an Intel processor:

- If, for example, the integer has a decimal value of 100 and the VARCHAR is ASCII '12345', then the input record appears in hexadecimal notation as:

```
X'64 00 00 00 05 00 31 32 33 34 35'
```

- If the data in the input record is in indicator mode, a single precedes the indicator byte:

```
X'00 64 00 00 00 05 00 31 32 33 34 35'
```

## Restrictions and Limitations

This section describes the restrictions and limitations when using the MultiLoad INMOD Adapter operator.

### Data Types

The operator cannot process these data types:

- BLOB
- CLOB
- JSON

- XML

# ODBC Operator

## ODBC Operator Capabilities

The ODBC operator is a producer operator that enables universal open data access to retrieve data from any non-Teradata ODBC-compliant data source. The ODBC operator is optimized and certified to work with Progress DataDirect, MS SQL Server, and Oracle native ODBC drivers. Attempting to use this operator with any other ODBC driver may have unexpected results, lack of functionality and possible decreased performance.

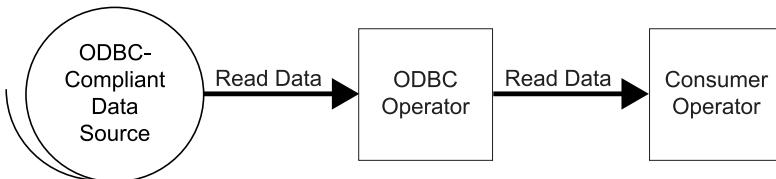
**Note:**

**Due to Progress DataDirect's lack of support for the MacOS platform, the ODBC operator is not officially supported on MacOS.**

Even though the ODBC operator can communicate with the Teradata RDBMS, it is more efficient to use either the Export operator or the SQL Selector operator to extract data from the Teradata RDBMS because these operators use the Teradata Call-Level Interface (CLlv2), which is more efficient. The Export operator is the fastest way to extract data from the Teradata RDBMS.

The ODBC operator also provides parallel access to different data sources. For example, in a single job script, you can use multiple ODBC operators to read from multiple data sources such as an Oracle server and a Microsoft SQL server. This is performed using the UNION ALL statement to combine the imported data.

The following figure shows the ODBC operator interface:



## Database Versions and Bundled DataDirect ODBC Drivers

The ODBC operator is certified with the following ODBC drivers, and requires that a copy of the following drivers be installed on the system from which the Teradata Parallel Transporter job is run.

DataDirect Driver Version	Oracle	Microsoft SQL Server	DB2	PostgreSQL	MySQL
<b>8.0.2</b>	12c, 18c, 19c	2012, 2014, 2016, 2017, 2019	9.1, 10.1, 11.5	8.2, 9.1, 10.1, 13.3, 14, 15	5.0, 8.0.26

## The ODBC Operator Function in a Teradata PT Job

When you use the ODBC operator in a Teradata PT job, Teradata PT directs one instance of the operator to do the following:

1. Log on to a SQL session with an ODBC-compliant data source.
2. Send the SELECT request specified in the operator attribute 'SelectStmt' to the data source.
3. Retrieve the data returned from the data source.
4. Send data to the Teradata PT data streams.
5. Log off the SQL session.
6. If the ODBC operator job is successful, terminate the job and report the total number of records exported from the database.
7. If the job is unsuccessful, terminate the job and provide information about the job so that you can correct any problem and restart the job.

The ODBC operator does the following:

- Accesses many ODBC-compliant data sources, for example, Oracle, SQL Server, DB2, and so on.
- Runs on all Teradata PT supported platforms
- Reads data close to the sources (from the same machine where Teradata PT is running, as opposed to across a network)
- Feeds data (using a data stream) directly to the database without the need of an intermediate staging platform.

## Syntax

### NOTICE

Use of this operator requires some knowledge of ODBC and how it works. For example, ODBC drivers use an ODBC initialization file (*odbc.ini*) to provide information on its connection to the data source and to specify data handling options. Using an initialization file makes it easy to configure unique options for your system. Therefore, you do not need to specify very many options manually when defining the ODBC operator. On UNIX systems, you can edit the initialization file directly. On Windows systems, a graphical interface is available.

Use the following specifications and attributes to the ODBC operator in a Teradata PT job script.

## Required Specifications

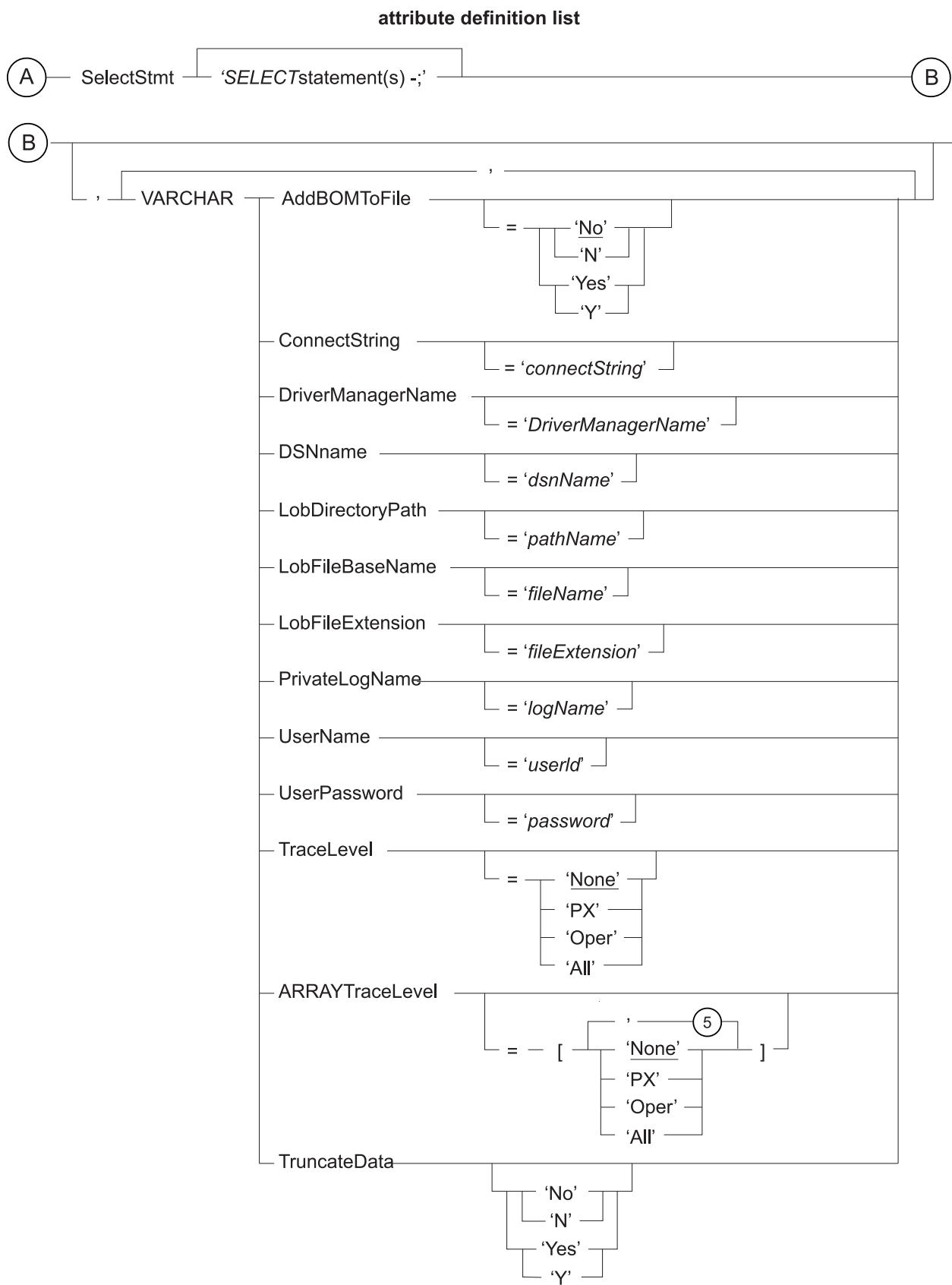
The following specifications are required to define a ODBC operator job.

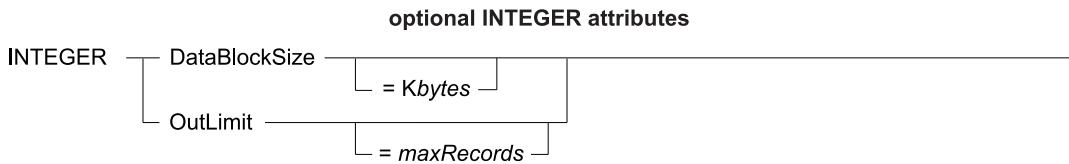
**Required Syntax for the ODBC Operator**

Specification...	Description
TYPE	Operator type. Always ODBC for the ODBC operator as a producer operator.
SelectStmt	Operator attribute that provides the Teradata SQL SELECT statement.

## Required and Optional Attributes

Use the attribute definition list syntax in the Teradata PT DEFINE OPERATOR statement to declare the required and optional attribute values for the ODBC operator.





where:

### ODBC Operator Attribute Definitions

Syntax Element...	Description
AddBOMToFile = 'option'	<p>Optional attribute that specifies whether the UTF byte-order-mark (BOM) will be added at the beginning of an XML, JSON, or CLOB output data file.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>'Y' or 'Yes' = The ODBC operator adds the appropriate UTF BOM at the beginning of an XML, JSON or CLOB output data file.</li> <li>'N' or 'No' = The ODBC operator does not prefix the appropriate UTF Byte-Order-Mark (BOM) at the beginning of an XML (JSON or CLOB) output data file (default).</li> </ul> <p>AddBOMToFile can only be used when following two conditions are met:</p> <p>The job schema has one or more XML, JSON, or CLOB columns defined AS DEFERRED BY NAME.</p> <p>The client character set for the load job is the Unicode character set.</p> <p>If these two conditions are not met, the values specified in the attribute are ignored.</p> <p><b>Note:</b></p> <p>The operator does not add a BOM to a data file extracted from a BLOB column in deferred mode.</p>
ConnectionString = 'connectString'	<p>Optional attribute that specifies an alternative method for connecting to the data source.</p> <p>If specified, the settings in this attribute supersede any settings in the DSNname, UserName, and UserPassword attributes. This string can contain any valid connect and/or driver information that is acceptable to the ODBC driver used.</p> <p>The settings in this attribute override the default settings in the ODBC initialization file.</p>
DataBlockSize = KBytes	<p>Optional attribute that allows you to fine tune ODBC operator performance by adjusting dynamically the data block size of the buffer that will hold multiple rows with a single fetch call.</p> <p>The size of the row affects the number of rows that can be fetched into that data block.</p> <p>For information on attribute limitations, see <a href="#">DataBlockSize</a>.</p>
DriverManagerName = 'DriverManagerName'	<p>Optional attribute that specifies the full path of the ODBC driver manager name.</p> <p>When the <i>DriverManagerName</i> attribute is not used, the default driver manager bundled with the Teradata Parallel Transporter is used.</p>

Syntax Element...	Description
	<p><b>Note:</b> The DriverManagerName attribute is not supported on z/OS and Windows. On Windows, the ODBC Operator uses the Windows default driver manager.</p>
DSNname = ' <i>dsName</i> '	<p>Optional attribute that specifies the name of the data source. Only the system data source name can be used. User data source name does not work. This name is also placed in the ODBC initialization file and can be used as a label in that file for providing connection information. The setting in this attribute overrides the default setting in the ODBC initialization file. The ConnectString attribute overrides this setting.</p>
LobDirectoryPath = ' <i>pathName</i> '	<p>Optional attribute that specifies the complete path name of an existing directory where all LOB, JSON, and XML data files will be written.</p>
LobFileBaseName = ' <i>fileName</i> '	<p>Optional attribute that defines a character string that will be prefixed to the names of LOB, JSON, and XML data files. The file names created by the ODBC operator are in the following format:</p> <pre>&lt;column-name&gt;_c&lt;#&gt;_&lt;job-id&gt;_p&lt;#&gt;_r&lt;#&gt;</pre> <p>where:</p> <ul style="list-style-type: none"> <li>• # that follows "c" is the column number.</li> <li>• # that follows "p" is the identification of the ODBC Operator copy.</li> <li>• # that follows "r" is the row order returned from the Database.</li> </ul> <p><b>Note:</b> The column names are sanitized to replace any character that is considered invalid for a Windows file name with an underscore '_'. These are the 9 characters that get replaced: \ / : * ? " &lt; &gt;   For consistency, the substitutions are done for UNIX platforms too. The '_c&lt;#&gt;' is added to ensure uniqueness of generated file names.</p> <p>For example, if you have the following schema:  <b>DEFINE SCHEMA &lt;schema-name&gt;</b></p> <pre>( COL2 BLOB AS DEFERRED BY NAME, COL3 CLOB AS DEFERRED BY NAME, COL4 XML AS DEFERRED BY NAME, COL5 JSON(1000000) AS DEFERRED BY NAME, );</pre> <p>where LobFileBaseName has the value "my_test", then the file names will be:</p> <ul style="list-style-type: none"> <li>• my_test_COL2_c1_&lt;job-id&gt;_p1_r1</li> <li>• my_test_COL3_c2_&lt;job-id&gt;_p1_r1</li> <li>• my_text_COL4_c3_&lt;job-id&gt;_p1_r1</li> <li>• my_text_COL5_c4_&lt;job-id&gt;_p1_r1</li> <li>• my_test_COL2_c1_&lt;job-id&gt;_p1_r2</li> <li>• my_test_COL3_c2_&lt;job-id&gt;_p1_r2</li> </ul>

Syntax Element...	Description
	<ul style="list-style-type: none"> <li>• my_test_COL4_c3_&lt;job-id&gt;_pl_r2</li> <li>• my_test_COL5_c4_&lt;job-id&gt;_pl_r2</li> <li>• my_test_COL2_c1_&lt;job-id&gt;_p1_r3</li> <li>• my_test_COL3_c2_&lt;job-id&gt;_p1_r3</li> <li>• my_test_COL4_c3_&lt;job-id&gt;_pl_r3</li> <li>• my_test_COL5_c4_&lt;job-id&gt;_pl_r3</li> </ul> <p>and so on.</p> <p>The files created will not have any extension unless specified in the LobFileExtension attribute.</p>
LobFileExtension = ' <i>fileExtension</i> '	<p>Optional attribute that specifies the extension for LOB, JSON, and XML data file names.</p> <p>Examples of 'file-extensions' include:</p> <ul style="list-style-type: none"> <li>• 'jpg': indicates that a file 'ccc.jpg' is a picture file.</li> <li>• 'gif': indicates that the file 'ddd.gif' is a picture file.</li> <li>• 'html': indicates that the file 'aaa.html' is an html file.</li> <li>• 'json': indicates that the file is a JSON file.</li> </ul>
OutLimit= <i>maxRecords</i>	<p>Optional attribute that specifies the maximum number of records processed by each instance of the operator.</p> <pre>INTEGER OutLimit = 1000</pre> <p>If specified, the OutLimit value must be greater than 0.</p> <p>If OutLimit is not specified, the number of records processed is not limited.</p> <p>The OutLimit specification applies to each instance of the Export operator and not to the total job.</p>
PrivateLogName = ' <i>logName</i> '	<p>Optional attribute that specifies the name of a log that is maintained by the Teradata PT Logger inside the public log. The private log contains all the output provided by the operator.</p> <p>The private log can be viewed using the tlogview command as follows, where <i>jobid</i> is the Teradata PT job name and <i>privatelogname</i> is the value for the operator's PrivateLogName attribute:</p> <pre>tlogview -j jobid -f privatelogname</pre> <p>If the private log is not specified, all output is stored in the public log.</p> <p>For more information about the tlogview command, see <a href="#">Teradata PT Utility Commands</a>.</p>
SelectStmt = 'SELECT <i>statements</i> ;'	<p>Required attribute that specifies a SQL SELECT statement that is sent to the ODBC-compliant data source. Row data is returned in the form of a result table.</p> <p>The operator does not parse the statement for validity. It is sent in its entirety without any type of processing.</p> <p>The SELECT statement can be any SQL SELECT statement whose syntax is supported by the database against which the ODBC operator connects.</p> <p>Not all databases support multistatement SELECTs. Therefore, because different data sources support different syntax, refer to the SQL reference for the specified ODBC-compliant data source, then enter only a valid SQL SELECT statement for the SelectStmt attribute.</p>

Syntax Element...	Description
TraceLevel = ' <i>level</i> '	<p>Optional attribute that specifies the types of diagnostic messages that are written by each instance of the operator to the public log (or private log, if one was specified using the PrivateLogName attribute). The diagnostic trace function provides more detailed information in the log file to aid in problem tracking and diagnosis.</p> <p>The valid trace levels are:</p> <ul style="list-style-type: none"> <li>• 'None' = TraceLevel turned off (default).</li> <li>• 'PX' = enables the tracing function for activities related to the Teradata PT infrastructure</li> <li>• 'Oper' = enables the tracing function for operator-specific activities</li> <li>• 'All' = enables tracing for all these activities</li> </ul> <p>The VARCHAR ARRAY can specify more than one value, for example:</p> <pre>VARCHAR TraceLevel = 'PX' VARCHAR TraceLevel = 'OPER' VARCHAR ARRAY TraceLevel = [ 'PX' ] VARCHAR ARRAY TraceLevel = [ 'PX', 'OPER' ]</pre> <p><b>Note:</b></p> <p>The TraceLevel attribute is provided as a diagnostic aid only. The amount and type of additional information provided by this attribute changes to meet evolving needs from release to release.</p>
TruncateData = ' <i>option</i> '	<p>Attribute that specifies whether data is to be truncated, when it arrives from the data source, to fit into the Teradata-type of column.</p> <p>This attribute is needed because databases other than Teradata (such as Oracle or DB2) might not use Teradata column types. For example, the maximum size of the DB2 column type LONG, which is equivalent to a VARCHAR column, is 2 MB. Teradata's maximum column size is 1MB. You can extract data from that column (just because it is defined as 2 MB in size does not mean the actual data is that large), but if the actual data exceeds 1MB, the ODBC operator needs to know whether to truncate that data or to stop with an error.</p> <p>The values are:</p> <ul style="list-style-type: none"> <li>• 'Yes' ( or 'Y') = truncate data.</li> <li>• 'No' ( or 'N') = do not truncate data (default).</li> </ul>
UserName = ' <i>userId</i> '	<p>Optional attribute that specifies the user name of the account or database in the data source.</p> <p>If not specified, then the ODBC driver looks in the initialization file for the user name information.</p> <p>The setting in this attribute overrides the default setting in the ODBC initialization file. The ConnectString attribute overrides this setting.</p>
UserPassword = ' <i>password</i> '	<p>Optional attribute that specifies the password associated with the UserName of the account or database in the data source.</p> <p>If not specified, then the ODBC driver will look in the ODBC initialization file for the user password information.</p> <p>The setting in this attribute will override the default setting in the ODBC initialization file. The ConnectString attribute will override this setting.</p>

## Usage Notes

### SelectStmt

When using the SelectStmt attribute to define the ODBC operator, multistatement SELECTs are allowed as long as the ODBC driver supports them.

The ODBC operator does not check the attribute for the presence of a multistatement SELECT.

If there is a multistatement SELECT, and the ODBC driver does not support it, the driver returns an error (although the error might not state that a multistatement SELECT was attempted) and the job terminates.

Also, the schema returned from the SELECT must match the schema as defined in the Teradata PT job script.



#### NOTICE

Oracle 9.2.0.1.0 incorrectly processes 1 MB SELECT statements, and might cause a loss of data. For more information, consult your Teradata field representative.

### DataBlockSize

Currently, formal testing using the ODBC Operator DataBlockSize attribute has been limited to 3 megabytes. This testing range produced a negative spike in performance that yielded longer elapse time values during the data row acquisition phase. As a result, you should consider, for example, row size, record size, number of columns, and so on to determine the optimal value to assign to the DataBlockSize attribute.

You should not set the job variable DataBlockSize less than the system default size of 1MB.

## Job Options

### Rules for Handling Decimal Data

The following rules apply to the ODBC Operator running on z/OS, UNIX, and Windows platforms, when taking DECIMAL data from a non-database system (such as Oracle) and moving it to a database system.

- If the precision values between the source and target tables are not equal, the data will be checked at runtime. If the data does not fit, the job will be terminated.
- If the scale factors do not match, one of the following will happen:
  - If the source scale factor is less than the target, then Teradata PT will add 0s (zeros) to the end of the source scale factor until they do match, and then the job will continue.
  - or,

- If the source scale factor is greater than the target, the job will terminate before attempting to process the data, due to possible loss of data.
- For source column NUMBER (m, n) and target column DECIMAL (M, N), the job will do the following:
  - Terminate with error if  $n > N$
  - Run if  $n < N$  or  $n = N$
  - Run if  $m \leq M$ , assuming  $n < N$  or  $n = N$
  - Attempt to run if  $m > M$ , assuming  $n < N$  or  $n = N$

## Using ODBC Operator with Oracle DATE Fields

The following limitations apply when the ODBC Operator runs on z/OS, UNIX, and Windows platforms.

- Whether the TruncateData flag is ON or OFF, the ODBC operator will truncate the Oracle Date to Vantage Date if the target column is mapped to INTDATE or ANSIDATE. If you don't want to truncate the Oracle Date then you must map the target column to TIMESTAMP or CHAR(19).
- Normal "DATE" fields in Oracle cannot copy correctly over to Vantage's DATE field (in ANSI mode).
- On some non-database systems, the DATE column type does not equate to Vantage's ANSI Date field. For example, on Oracle, the column type DATE is equivalent to the column type DATETIME, which has a format of YYYY-MM-DD HH:MM:SS (equivalent to a CHAR(19)). Vantage ANSI DATE has a format of YYYY-MM-DD (equivalent to CHAR(10)).

## Support for NUMBER Data Type

When extracting NUMBER(m,n), the target column can be defined as NUMBER (M,N).

The following conditions still hold in the Teradata PT job for the values m and n.

A Teradata PT job:

- Terminates with error if  $n > N$
- Runs if  $n < N$  or  $n = N$
- Runs if  $m \leq M$ , assuming  $n < M$  or  $n = N$
- Attempts to run if  $m > M$ , assuming  $n < N$  or  $n = N$

## Configuring ODBC Initialization Files

Before using the ODBC operator, you must configure the ODBC initialization file on your platform to designate the DSN (data source name) in 'ODBC Data Sources' to point to the proper data source.

Following is a sample ODBC.ini file for a Solaris client system. It shows data source information for Teradata, ORACLE, and SQLServer data sources.

```
[ODBC]
InstallDir=/usr/odbc
Trace=0
```

```

TraceFile=/usr/joe/odbcusr/trace.log
TraceAutoStop=0

[ODBC Data Sources]
testdsn=tdata.so
default=tdata.so
Teradata=tdata.so
Oracle=DataDirect 4.10 Oracle
SQLServer Wire Protocol=DataDirect 4.10 SQL Server Wire Protocol

[testdsn]

[default]

[Teradata]

[Oracle]
Driver=/opt/odbc/lib/ivor818.so
Description=DataDirect 4.10 Oracle
LogonID=oracle
Password=oracle
ServerName=DBCName=200.001.001.01
CatalogOptions=0
ProcedureRetResults=0
EnableDescribeParam=0
EnableStaticCursorsForLongData=0
ApplicationUsingThreads=1

[SQLServer Wire Protocol]
Driver=/opt/odbc/lib/ivmsss18.so
Description=DataDirect 4.10 SQL Server Wire Protocol
Database=db
LogonID=uid
Password=pwd
Address=sqlserverhost,1433
QuotedId=No
AnsiNPW=No

```

## Prerequisites

Because using the ODBC operator in a Teradata PT job script requires the following:

- A copy of the ODBC.INI (initialization) file has been installed on your Client platform, and

- The ODBC.INI (initialization) file has been configured on your Client platform to specify the:
  - DSNName
  - HostName
  - ServiceName.

This enables your Teradata PT script (through the DSNName in the ODBC.INI file and the same DSNName attribute for the ODBC operator) to know that name of the system on which the Oracle database resides (identified in the ODBC.INI file by the Oracle HostName and ServiceName).

For the procedure for configuring the ODBC.INI file on your Client platform with the DSNName and the Oracle HostName and ServiceName.

## Using the ODBC Operator on IBM z/OS

### Ensuring that the HOST ODBC Driver is Set Up

This task is usually performed by a z/OS system programmer or DB2 DBA.

The Mainframe HOST DB2 ODBC interface uses a plan named DSNACLI. Part of the DB2 installation process involves binding this plan and its associated packages (see member DSNTIJCL in SDSNSAMP).

For more information, see the *DB2 z/OS ODBC Guide and Reference*.

**Note:**

The mentioned interface is not the JDBC interface. Some windows DESKTOP software, including many IBM applications use JDBC. Just because your windows software is functional does not mean the mentioned task has been performed correctly.

### Ensuring that UNIX User IDs are Set Up

This task is performed by the system security staff or system programmers.

Teradata PT requires that User OMVS Segments are defined.

- The userId must be correctly set up for USS (UNIX) access. In z/OS terms, this is called adding an OMVS segment with a valid (UID/GID) and (home dir).

**Note:**

If the OMVS segment is not defined, the results are undetermined.

Teradata PT hangs when the default OMVS Segment is used because IBM (when running the default OMVS segment) does not support some of the callable services used by Teradata PT, among them kill(), pidaffinity(), trace(), and sigqueue().

- The userId must have correct RACF/ACF2/Top Secret security authorizations to the DBMS and the database in question.

## Obtaining Database System Identity Information

The following information, used by the Teradata PT ODBC operator, must be provided by a DB2 DBA or z/OS system programmer:

- Because the IBM DB2 database is a subsystem, knowing the subsystem ID (SSID) is required.
- Knowing the data source name of the database, referred to as the “connection name” or the “location name,” is in most cases required. The connection or location name is defined in SYSIBM.LOCATIONS when DDF is configured in the DB2 subsystem.

In many applications, when DDF is not being used, a local database is accessed. In these cases, the local database name is the name that was set during DB2 installation as DB2 LOCATION NAME on the DSNTIPR installation panel for the DB2 subsystem.

When the data source name is not provided, the default local database will be used. The local database is identified by the SSID.

## Setting Up Teradata PT Job JCL for ODBC

To define a subsystem to DB2 ODBC:

- Specify the MVSDEFAULTSSID keyword in the common section of the initialization file identified by the DDNAME “DSNOAINI”.
- If the MVSDEFAULTSSID keyword does not exist in the initialization file, DB2 ODBC uses the default subsystem name specified in the DSNHDECP load module that was created when DB2 was installed.

The DSNHDECP load module is usually link-edited into the \*.SDSNEXIT data set. See the following example:

```
//STEPLIB DD DSN=DSN910.SDSNEXIT,DISP=SHR
//          DD DSN=DSN910.SDSNLOAD,DISP=SHR
```

To set up a Teradata PT Job JCL for ODBC:

- Add the DB2 library to the JOBLIB/STEPLIB, as in the following example:

```
//JOBLIB DD ( ... )
//          DD DSN=DB2910.SDSNLOAD,DISP=SHR
```

- (Optional) Add the DB2 Initialization Dataset with the Teradata PT jobstep/procstep override, as in the following example:

```
//TPT.DSNAOINI DD DSN=DB2910.INIT(DB2INIT)
```

```
DB2INIT:  
[COMMON]  
MVSDEFAULTSSID=SSID
```

## Updating the Teradata PT Job Variables File

Set the following variables in the job variables file:

```
Data_Source = 'connection-name'  
DB2UserName = 'johndoe'  
DB2Password = 'abcd1234'
```

## Updating the Teradata PT Job Script

Define the following attributes in the ODBC job script:

```
VARCHAR DSNNName      = @Data_Source,  
VARCHAR UserName       = @DB2UserName,  
VARCHAR UserPassword   = @DB2Password,
```

---

### Note:

The DSNNName is the data source name. When this is not provided, the default local database identified by the SSID is used. When the user name and password are not provided the RACF credentials of the job and/or the name of the user who submitted the job are used.

---

## Debugging Connection Errors

To debug connection errors, set the ODBC Operator script "tracelevel" to "special" in the ODBC job script, as follows:

```
VARCHAR TraceLevel = 'Special'
```

The following are common connection errors:

- When the DSNACLI plan is not bound at DB2 installation time, the following error occurs:

```
Fatal error received from ODBC driver:  
STATE=58004, CODE=-99999,
```

```

MSG='DB2 for OS/390}{ODBC Driver}
SQLSTATE=58004  ERRLOC
CAF "OPEN" failed using DB2 system:DSN9 and PLAN:DSNACLI
RC=0c and REASON=00f30040

```

**Note:**

After maintenance is applied to DB2, a connection failure of (-803) indicates that the plan needs to be rebound.

- When you have not obtained DB2 and RACF (ACF2) authorization, the following error occurs:

```

Fatal error received from ODBC driver:
STATE=42505, CODE=-922,
MSG='DB2 for OS/390}{ODBC Driver}
DSNT408I SQLCODE = -922, ERROR: AUTHORIZATION FAILURE: 00D31024    ERROR.
CONNECT
DSNT418I SQLSTATE    = 42505 SQLSTATE RETURN CODE

```

- If ODBC does not connect, the following error occurs:

```

Fatal error received from ODBC driver:
STATE=58004, CODE=-99999,
MSG='DB2 FOR OS/390}{ODBC DRIVER}  SQLSTATE=58004  ERRLOC=2:170:9
CAF "CONNECT" failed using DB2 system:DSN9
RC=08 and REASON=00f30002

```

This can occur for two reasons:

- The Data Source Name is incorrect or does not exist.
- The database system is not running.

## Using the ODBC Operator on Unix Platforms

Observe the following requirements when using the ODBC operator on Unix platforms:

- When using the attribute DriverManagerName in the ODBC operator to use your own driver manager instead of the bundled driver manager, you must set and export the LD\_LIBRARY\_PATH (or platform equivalent) environment variable to the path where your driver manager shared libraries are located.
- When using the bundled Progress DataDirect driver manager with the ODBC operator, TPT will automatically set the LD\_LIBRARY\_PATH environment variable to the correct location. Do not set the attribute DriverManagerName in the ODBC operator.

## Using Teradata-Provided Branded Drivers

Teradata will distribute the Teradata-branded Progress DataDirect ODBC Drivers.

**Note:**

The branded drivers are provided for Oracle, SQL Server, DB2, PostgreSQL, and MySQL only, and using these branded ODBC Drivers requires that a valid license copy be placed with the ODBC drivers, or these drivers will continue to display lengthy warning messages when a job is run. The license to use the branded drivers must be requested from Teradata by your site team. You cannot use your own private Progress DataDirect license for these branded drivers. You are not allowed to use the license and these branded drivers for any other product except for Teradata Parallel Transporter.

---

If you do not have a Progress DataDirect ODBC driver permanent license, contact your Teradata Account Representative or Teradata Customer Support to procure a permanent license for the bundled drivers. Alternatively, you can order the DataDirect Driver Connector license and drivers from Progress Software. For more information, refer to <https://www.progress.com>.

## **Location of the Branded ODBC Drivers**

The Teradata-branded ODBC drivers are located under the TPT Installation Directory in a folder named "odbc". This folder contains 64-bit drivers, where "lib64" points to the 64-bit drivers.

The ODBC.INI (initialization) file for 64-bit drivers is located in the odbc folder of the TPT install directory. For information about configuring the ODBC initialization file, see [Configuring ODBC Initialization Files](#).

## **Behavior of the ODBC Driver when Using/Not Using these Drivers**

### **Case 1: Using the Branded Drivers with a Valid License File Provided by Teradata**

The TPT job runs successfully and extracts the records as expected.

### **Case 2: Using the Branded Drivers without a Valid License File Provided by Teradata**

The TPT job terminates and the following warning is displayed on the console:

```
ODBC_OPERATOR: TPT17101: Fatal error received from ODBC driver:  
STATE=HY000, CODE=6060,  
[TPT][ODBC Oracle Wire Protocol driver]You are not licensed to use this  
DataDirect Technologies product under the license you have purchased.  
If you wish to purchase a license for use with this application, then you may  
use this product for a period of 15 days, during which time you are required to  
obtain a license.
```

[TPT][ODBC Oracle Wire Protocol driver] You can order a license for a DataDirect Technologies product for use with this application by calling DataDirect Technologies at 800-876-3101 in North America and +44 (0) 1753-218 930 elsewhere. Thank you for your cooperation.

**Note:**

The mentioned case on Windows (Case 2) results in a pop-up warning message box and waits for user input. It is not advisable to execute batch jobs using ODBC operator jobs without the proper license file.

### Case 3: Using any drivers other than the ones provided by Teradata

The TPT job runs successfully and displays the following warning on the console:

ODBC\_OPERATOR: TPT17199: Warning: The ODBC Driver used is not distributed by Teradata and might give unexpected results.

**Note:**

The DataDirect branded drivers are not available on the MacOS platform.

## Operational Considerations

### Checkpointing and Restarting

The ODBC operator does not support checkpoint and restart operations because it is unknown how the databases it can connect to handle restarts.

### TD Wallet

The ODBC Operator supports TD Wallet with the following restrictions:

- It is supported only in the DSNName, UserName, UserPassword, and ConnectString attributes of the ODBC Operator.
- The syntax supported is limited to '\$tdwallet(<key>)' where '\$tdwallet' is to be specified in lower case only. And the value for the <key> should always begin with 'tptodbcoperator\_' as its prefix. If not, the requested key will not be retrieved by the TPT ODBC Operator and a user error is reported. The prefix 'tptodbcoperator\_' is case insensitive.

For example, if a key named 'tptodbcoperator\_MyPassword' has a value in the wallet as 'abcPassword' and the Operator definition is given as:

```
VARCHAR UserPassword = '$tdwallet(tptodbcoperator_MyPassword)'
```

the string '\$tdwallet(tptodbcoperator\_MyPassword)' will be replaced during the job execution as:

```
VARCHAR UserPassword = 'abcPassword'
```

No other combinations are supported.

- If any other character is added to the TD Wallet calling syntax, the total string is treated as the attribute value.

For example, if in the operator definition, the UserName is defined as:

```
VARCHAR UserName = 'xxx$tdwallet(tptodbcoperator_Myusername)'
```

The string is not looked up in the TD Wallet and the total string 'xxx\$tdwallet(tptodbcoperator\_Myusername)' is treated as the UserName.

- There is also a support for recursive key values to an extent of 15 recursions only, and in case of excess recursions, the job terminates with an error.

For example if a key named 'tptodbcoperator\_MyPassword' has a value in the wallet as '\$Stdwallet(tptodbcoperator\_user1)' and the value in the Wallet for 'tptodbcoperator\_user1' is stored as 'user1password', and the operator definition is given as:

```
VARCHAR UserPassword = '$tdwallet(tptodbcoperator_MyPassword)'
```

the string '\$tdwallet(tptodbcoperator\_MyPassword)' will be replaced during the job execution as:

```
VARCHAR UserPassword = 'user1password'
```

with 1 level of recursion.

- In case the string used as the key to the value Teradata Wallet is not found, this results in the job termination.
- When using the recursive strings for the TD Wallet support in the ODBC Operator, it is mandatory to have the resultant recursive values also have the key starting with 'tptodbcoperator\_' except the last value, which you actually intend to use as the value for that ODBC Operator attribute.
- When using the TD Wallet feature with the ConnectionString attribute, the value corresponding to the key provided should result in a complete or a minimal acceptable ConnectString, with which the ODBC Operator can establish a connection with the DataSource.
- TD Wallet is not supported on the z/OS platform in the ODBC Operator.

## LOB/CLOB Support

The ODBC Operator supports LOB/CLOB types. Currently the ODBC Operator is certified to extract LOB/CLOB columns from Oracle, MS SqlServer, DB2, PostgreSQL, and MySql only.

**Note:**

The ODBC Operator does not support LOB/CLOB types on z/OS.

## LOB Extraction Attributes

The following attributes are only for extracting LOB, JSON, or XML data in deferred mode:

- LobDirectoryPath
- LobFileName
- LobFileExtension
- AddBOMToFile

That is, when LOB columns are defined as follows in a schema:

- BLOB[length] AS DEFERRED BY NAME
- CLOB[length] AS DEFERRED BY NAME
- JSON [length] AS DEFERRED BY NAME
- XML AS DEFERRED BY NAME

## Operational Considerations

- Though the ODBC operator supports multiple instances, if the schema has at least one deferred mode LOB column defined, the ODBC Operator uses only one instance to do the data fetch. So for better performance with the LOB types, it is advised to use only one instance.
- The ODBC operator also supports the usage of multiple LOB columns in the schema. Also you can use a combination of inline and deferred columns in the schema.
- The ODBC operator supports inline LOBs up to the max size for the LOB.
- The ODBC operator supports enhanced LOB functionality.

For more information, see [Enhanced LOB Support](#).

## Restrictions and Limitations

The restrictions and limitations of ODBC Operator are listed here:

- A BINARY\_FLOAT column in a table on an Oracle database cannot be exported and loaded to an Analytics Database FLOAT column. This is because a BINARY\_FLOAT column is a 4-byte floating-point value and the Analytics Database does not support a 4-byte floating-point column. Analytics Database only supports a 8-byte floating-point column.
- The bundled DataDirect ODBC drivers do not support INTERVAL data types.
- Trailing blanks in CHAR column on MySQL are removed during the retrieval when the column is defined with Unicode character set.

For example, the column is defined as follows:

- `col_char CHAR(10) CHARACTER SET UTF16`

It has the value 'abc', which is stored as 'abc ' (with 7 blanks) in the table. When the column is retrieved, its value is 'abc' and the operator stores the value in the buffer of 10 bytes as 'abc\0\0\0\0\0\0\0\0' (with 7 '\0' following the actual value).

The value with padded NULL characters has inserted a Teradata CHAR column; that alters the column value in a target table.

Teradata recommends using VARCHAR instead of CHAR. If you use RPAD on the CHAR column in the SQL SELECT statement, the returned SQL data types for the column is SQL\_WVARCHAR. The ODBC operator then processes it as VARCHAR format, instead of CHAR format. That leads to a failure while loading it into a CHAR column in a Teradata table.

# OS Command Operator

## OS Command Operator Capabilities

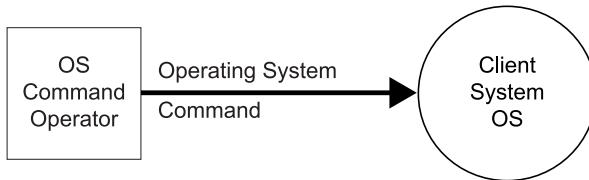
OS Command is a standalone operator that executes operating system commands on the client system that runs the job.

Any output from executed commands is written to the job log.

For Teradata PT jobs running on z/OS, the OS Command operator has limited functionality. It cannot execute z/OS system commands, although it can execute USS commands.

The following figure shows the OS Command operator interface.

### ***OS Command Operator Interface***



## The OS Command Operator Function in a Teradata PT Job

When you use the OS Command operator in a Teradata PT job, Teradata PT directs one instance of the operator to:

1. Spawn an OS command.
2. Read the output from the OS command.
3. Send the output to the Teradata PT Logger.

## Syntax

### **Note:**

It is assumed that you possess some knowledge of your operating system and how it works.

Use the following specifications and attributes to define the OS Command operator in a Teradata PT job script.

## Required Specifications

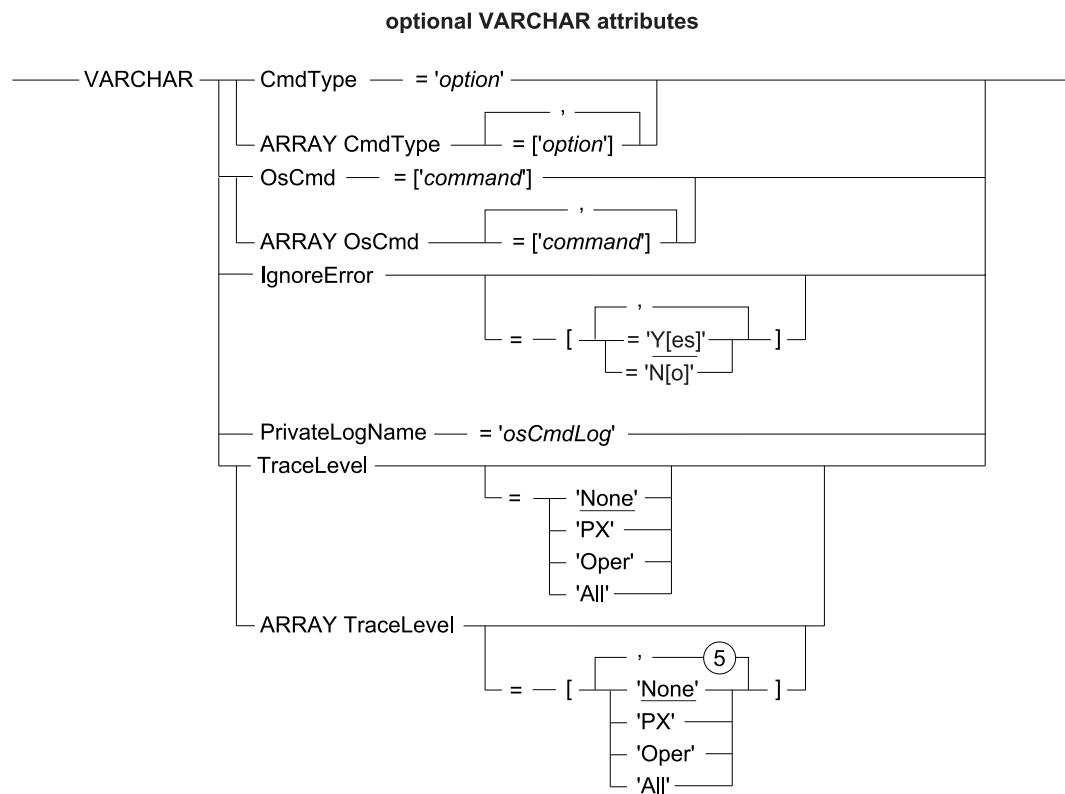
The following specifications are required to define the OS Command operator in a Teradata PT job script.

### Required Syntax for the OS Command Operator

Specification	Description
OsCmd = ' <i>command</i> '	Name of the operating system command. To specify more than one command, use the following syntax:  VARCHAR ARRAY OsCmd = ['cmd1', 'cmd2', 'cmd3'...]  Each command is executed in a separate call to the operating system.
TYPE	Operator type. Always OS COMMAND for the OS Command operator.

## Required and Optional Attributes

Use the attribute definition list syntax in the Teradata PT DEFINE OPERATOR statement to declare the required and optional attribute values for the OS Command operator.



where:

### OS Command Operator Attribute Descriptions

Syntax Element	Description
CmdType='option'	<p>Optional attribute that specifies the command type (only for IBM z/OS). Valid values are:</p> <ul style="list-style-type: none"> <li>• 'tso'</li> <li>• 'uss'</li> </ul> <p>If nothing is specified, 'uss' is the default.</p> <p>The argument can be a single VARCHAR ('tso' or 'uss') or an array VARCHAR ([‘tso’, ‘uss’]).</p> <p>The following is invalid: [, ‘uss’, , ‘tso’].</p>
IgnoreError = 'option'	<p>Optional attribute that specifies whether to ignore an error returned from the operating system or to terminate.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>• 'Yes' (or 'Y') = continue the Teradata PT job when an error returns from the execution of the operating system execution (default).</li> <li>• 'No' (or 'N') = terminate the Teradata PT job when an error returns from the execution of the operating system.</li> </ul> <p>The VARCHAR ARRAY can specify more than one value. For example:</p> <ul style="list-style-type: none"> <li>• VARCHAR IgnoreError = 'YES'</li> <li>• VARCHAR IgnoreError = 'NO'</li> <li>• VARCHAR ARRAY IgnoreError = [ 'YES' ]</li> <li>• VARCHAR ARRAY IgnoreError = [ 'YES', 'NO' ]</li> </ul> <p>When used as in this case, the condition corresponds to the number of the command used in the ARRAY OsCmd attribute.</p>
PrivateLogName = 'logName'	<p>Optional attribute that specifies the name of a log that is maintained by the Teradata PT Logger inside the public log. The private log contains all the output provided by the operator.</p> <p>The private log can be viewed using the tlogview command as follows, where jobId is the Teradata PT job name and <i>privateLogName</i> is the value for the operator's PrivateLogName attribute:</p> <pre>tlogview -j jobId -f privateLogName</pre> <p>If the private log is not specified, all output is stored in the public log.</p> <p>For more information about the tlogview command, see <a href="#">Teradata PT Utility Commands</a>.</p>
TraceLevel = 'level'	<p>Optional attribute that specifies the types of diagnostic messages that are written by each instance of the operator to the to the public log (or private log, if one is specified using the PrivateLogName attribute). The diagnostic trace function provides more detailed information in the log file to aid in problem tracking and diagnosis.</p> <p>The trace levels are:</p> <ul style="list-style-type: none"> <li>• 'None' = TraceLevel turned off (default).</li> <li>• 'PX' = enables the tracing function for activities related to the Teradata PT infrastructure</li> <li>• 'Oper' = enables the tracing function for operator-specific activities</li> <li>• 'All' = enables tracing for all the these activities</li> </ul>

Syntax Element	Description
	<p>The VARCHAR ARRAY can specify more than one value, for example:</p> <pre>VARCHAR TraceLevel = 'PX' VARCHAR TraceLevel = 'OPER' VARCHAR ARRAY TraceLevel = [ 'PX' ] VARCHAR ARRAY TraceLevel = [ 'PX', 'OPER' ]</pre> <p><b>Note:</b> The TraceLevel attribute is provided as a diagnostic aid only. The amount and type of additional information provided by this attribute changes to meet evolving needs from release to release.</p>
VARCHAR	Keyword that specifies VARCHAR as the data type of the defined attribute.

## Usage Notes

### Restrictions and Limitations

The OS Command operator is neither a producer nor consumer operator.

- The OS Command operator is a standalone operator.
- The OS Command operator uses one instance. It does not support multiple instances.
- The OS Command operator can execute USS commands, but it cannot execute z/OS system commands.
- By default, if the OS Command operator encounters an error from the execution of the operating system, the OS Command operator will ignore the error and continue the Teradata PT job. You can override this behavior by using the OS Command operator's IgnoreError attribute.

# Schema Mapping Operator

## Schema Mapping Operator Capabilities

The Schema Mapping operator displays the data sent by the producer operator according to the user-provided attributes and the format specified in the DEFINE SCHEMA statement of a Teradata PT job. Output is written to the Teradata PT private log.

Use this operator to:

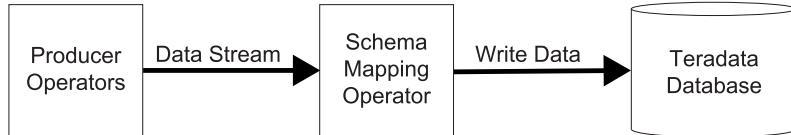
- Enable the user to verify that the schema definition correctly describes the input data.
- Allow data to be displayed in various formats for debugging purposes.

All Schema Mapping operator output is written to the Teradata PT private log.

All platforms on which Teradata PT is supported support the Schema Mapping operator.

The following figure shows the Schema Mapping operator interface.

### ***Schema Mapping Operator Interface***



## Syntax

Use the following specifications and attributes to define the Schema Mapping operator in a Teradata PT job script.

## Required Specifications

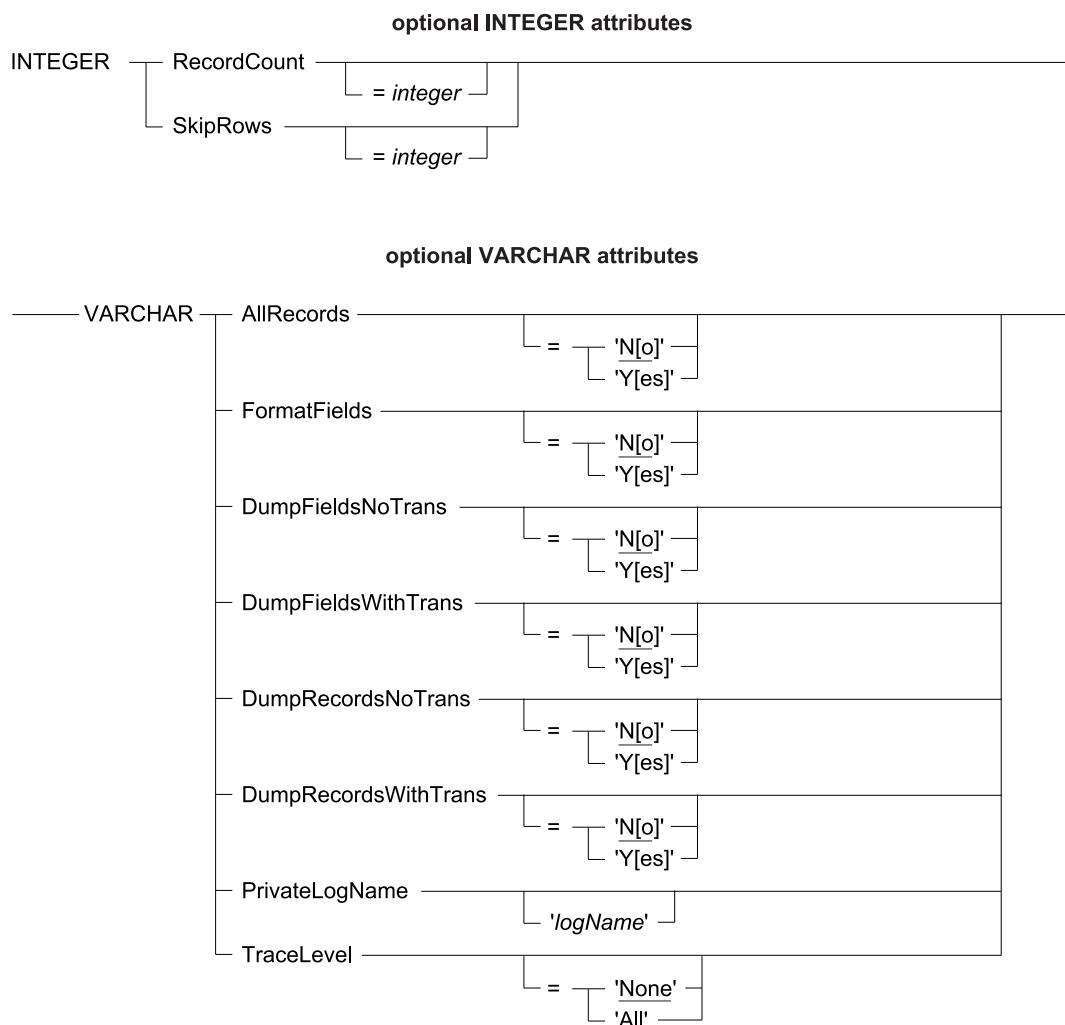
The following specifications are required to define a Schema Mapping operator.

### **Required Syntax for the Schema Mapping Operator**

Specification	Description
TYPE	Operator type. Always SCHEMAMAPPER for the Schema Mapping operator.

## Optional Attributes

Use the attribute definition list syntax in the Teradata PT DEFINE OPERATOR statement to declare the optional attribute values for the Schema Mapping operator.



where:

### Schema Mapping Operator Attribute Descriptions

Syntax Element	Description
AllRecords ='option'	<p>Optional attribute that specifies whether to display all records whose data format conforms to the data format specified in the DEFINE SCHEMA statement for the Teradata PT schema mapping job.</p> <p>Valid options are:</p> <ul style="list-style-type: none"> <li>• 'No' = display 1 records (default).</li> <li>• 'Yes' = display all records.</li> </ul>

Syntax Element	Description
FormatFields = ' <i>option</i> '	<p>Optional attribute that specifies whether to display the hexadecimal offset of record fields and the record fields themselves in hexadecimal format from the beginning of the field.</p> <p>FormatFields does not display control bytes. For example, FormatFields does not display the end-of-line or end-of-file marker byte.</p> <p>Valid options:</p> <ul style="list-style-type: none"> <li>‘No’ = do not display this format (default).</li> <li>‘Yes’ = display the hexadecimal offset of record fields (by record field) and the record fields in hexadecimal.</li> </ul>
DumpFieldsNoTrans = ' <i>option</i> '	<p>Optional attribute that specifies whether to display fields in hexadecimal format, but not to display the transformation of hexadecimal characters into ASCII characters.</p> <p>Valid options:</p> <ul style="list-style-type: none"> <li>‘No’ = do not display this format unless all the other format display options are set to ‘No’ (default).</li> <li>‘Yes’ = display fields in hexadecimal characters but not ASCII characters.</li> </ul>
DumpFieldsWithTrans = ' <i>option</i> '	<p>Optional attribute that specifies whether to display fields in hexadecimal format, including the transformation of hexadecimal characters into ASCII characters.</p> <p>Valid options:</p> <ul style="list-style-type: none"> <li>‘No’ = do not display this format (default).</li> <li>‘Yes’ = display fields in hexadecimal characters as well as ASCII characters.</li> </ul>
DumpRecordsNoTrans = ' <i>option</i> '	<p>Optional attribute that specifies whether to display records in hexadecimal format, but not to display the transformation of hexadecimal characters into ASCII characters.</p> <p>Valid options:</p> <ul style="list-style-type: none"> <li>‘No’ = do not display this format (default).</li> <li>‘Yes’ = display records in hexadecimal format but not ASCII characters.</li> </ul>
PrivateLogName = ' <i>option</i> '	<p>Optional attribute that specifies the name of a log that is maintained by the Teradata PT Logger inside the public log. The private log contains all the output provided by the operator.</p> <p>The private log can be viewed using the tlogview command as follows, where jobid is the Teradata PT job name and privateLogName is the value for the operator's PrivateLogName attribute:</p> <pre>tlogview -j jobid -f privatelogname</pre> <p>If the private log is not specified, all output is stored in the public log. By default, no diagnostic trace messages are produced. Diagnostic trace messages are produced only when the user sets a valid value for the TraceLevel attribute.</p> <p>For more information about the tlogview command, see <a href="#">Teradata PT Utility Commands</a>.</p>
DumpRecordsWithTrans = ' <i>option</i> '	<p>Optional attribute that specifies whether to display records in hexadecimal format, including the transformation of hexadecimal characters into ASCII characters.</p>

Syntax Element	Description
	<p>Valid options:</p> <ul style="list-style-type: none"> <li>‘No’ = do not display this format (default).</li> <li>‘Yes’ = display records in hexadecimal characters as well as ASCII characters.</li> </ul>
RecordCount = <i>integer</i>	<p>Optional attribute that indicates the number of records to be displayed whose data format conforms to the user-provided attributes and the data format specified in the DEFINE SCHEMA statement for the Teradata PT schema mapping job.</p> <p>Valid values: any valid positive integer from 0 to 4294967296.</p>
SkipRows= <i>integer</i>	<p>Optional attribute that indicates the number of rows to be skipped before starting to display records whose data format conforms to the user-provided attributes and the data format specified in the DEFINE SCHEMA statement for the Teradata PT schema mapping job.</p> <p>For example, if RecordCount is set to 10 and SkipRows is set to 5, then 5 rows are skipped and the Schema Mapping operator displays the last 5 records and the record numbers would be 6 to 10.</p> <p>Valid value: any positive integer.</p> <p>Default value for SkipRows is 0. This means no records will be skipped.</p>
TraceLevel = ‘level’	<p>Optional attribute that specifies the types of diagnostic messages that are written by each instance of the operator to the public log (or private log, if one is specified using the PrivateLogName attribute). The diagnostic trace function provides more detailed information in the log file to aid in problem tracking and diagnosis. The trace levels are:</p> <ul style="list-style-type: none"> <li>‘None’ = TraceLevel turned off (default).</li> <li>‘All’ = enables tracing for all the mentioned activities</li> </ul> <p><b>Note:</b></p> <p>The TraceLevel attribute is provided as a diagnostic aid only. The amount and type of additional information provided by this attribute changes to meet evolving needs from release to release.</p>

## Attribute Relationships

The following table illustrates the relationship between Schema Mapping operator attributes DumpFieldsNoTrans and DumpFieldsWithTrans.

**Relationship Between DumpFieldsNoTrans and DumpFieldsWithTrans**

If Attribute...	Is Set To.. ..	And Attribute...	Is Set To.. ..	Schema Mapping Operator Returns...
DumpFieldsNoTrans	Y	DumpFieldsWithTrans	N	DumpFieldsNoTrans
DumpFieldsNoTrans	N	DumpFieldsWithTrans	Y	DumpFieldsWithTrans
DumpFieldsNoTrans	Y	DumpFieldsWithTrans	Y	DumpFieldsWithTrans

If Attribute...	Is Set To... ..	And Attribute...	Is Set To... ..	Schema Mapping Operator Returns...
DumpFieldsNoTrans	N	DumpFieldsWithTrans	N	no output for these attributes

The following table illustrates the relationship between Schema Mapping operator attributes DumpRecordsNoTrans and DumpRecordsWithTrans.

#### Relationship Between DumpRecordsNoTrans and DumpRecordsWithTrans

If Attribute...	Is Set To...	And Attribute...	Is Set To...	Schema Mapping Operator Returns...
DumpRecordsNoTrans	Y	DumpRecordsWithTrans	N	DumpRecordsNoTrans
DumpRecordsNoTrans	N	DumpRecordsWithTrans	Y	DumpRecordsWithTrans
DumpRecordsNoTrans	Y	DumpRecordsWithTrans	Y	DumpRecordsWithTrans
DumpRecordsNoTrans	N	DumpRecordsWithTrans	N	no output for these attributes

The following table illustrates the relationship between the Schema Mapping operator attribute FormatFields and the “dump” attributes.

FormatFields is independent of all “dump” attributes except when FormatFields and all “dump” attributes are set to N.

#### Relationship Between FormatFields and the Dump Attributes

If Attribute...	Is Set To...	And Attribute...	Is Set To...	Schema Mapping Operator Returns...
FormatFields	Y	none	none	FormatFields
	N	DumpRecordsNoTrans DumpRecordsWithTrans DumpFieldsNoTrans DumpFieldsWithTrans	N	DumpFieldsNoTrans

The following table illustrates the relationship among the Schema Mapping operator attributes RecordCount and AllRecords:

#### Relationship Between RecordCount and AllRecords

If Attribute...	Is Set To...	And Attribute...	Is Set To...	Schema Mapping Operator Returns...
RecordCount	zero	AllRecords	N	1 record
RecordCount	non-zero	AllRecords	N	RecordCount number
RecordCount	zero	AllRecords	Y	all records
RecordCount	non-zero	AllRecords	Y	RecordCount number

## Schema Mapping Output Examples

### DumpFieldsNoTrans

DumpFieldsNoTrans set to Y produces the following output in the Teradata PT private log:

```
Record Number : 19
COL001      38 36 36 30 2D 31 31 2D 30 31
COL003      05
COL005      53
COL006      54
COL007      55
COL009      FF FE C6 46
COL012      80 00 56 FD
COL015      00 01 00 00 00
```

The output shows:

- The record number
- The name of the field
- The field itself in hexadecimal format

### DumpFieldsWithTrans

DumpFieldsWithTrans set to Y produces the following output in the Teradata PT private log:

```
Record Number : 19
COL001      (ANSIDATE)      8660-11-01
COL003      (BYTEINT)       05
COL005      (CHAR(1))        S
COL006      (CHAR(1))        T
COL007      (CHAR(1))        U
COL009      (DECIMAL(5,2))   -803.14
COL012      (INTEGER)        -2147461379
COL015      (VARBYTE(5))     00
```

The output shows:

- The record number
- The name of the field
- The data type of the field
- The field itself in ASCII format

## DumpRecordsNoTrans

DumpRecordsNoTrans set to Y produces the following output in the Teradata PT private log:

```
Record Number : 19
DUMPRECORDNOTRANS: max bytes, reclen: 1024, 26
    Hexadecimal formatted display from address FE18272A for 26 bytes.
0000      38 36 36 30.2D 31 31 2D.30 31 05 53.54 55 FF FE
0010      C6 46 80 00.56 FD 00 01.00 00 00
```

The output shows:

- The record number
- The name of the attribute output that has been returned
- The record length
- Address of the record
- The record itself in hexadecimal format

## DumpRecordsWithTrans

DumpRecordsWithTrans set to Y produces the following output in the Teradata PT private log:

```
Record Number: 19
max bytes, reclen: 1024, 26
0000 38 36 36 30.2D 31 31 2D.30 31 05 53.54 55 FF FE  8660-11-01.STU../*
0010 C6 46 80 00.56 FD 00 01.00 00 00 .F..V.....*
```

The output shows:

- The record number
- The name of the attribute used for displaying the data format
- The record length
- Address of the record
- The record itself in hexadecimal format
- The record itself in ASCII characters

## FormatFields

FormatFields set to a value of Y produces the following output in the Teradata PT private log:

```
Record Number: 19
Field1      0000      0000CAFE
Field2      0004      00000BAD
```

Field3	0008	5758595A
Field4	000C	32303030
Field5	0010	2D30312D 3031
Field6	0016	001B5465 72616461 7461

The output shows:

- The record number
- The name of the field
- The hexadecimal offset of the field. For example, field2 starts with offset x'4' and field5 start with offset x'16'
- The field itself in hexadecimal characters from the beginning of the field

FormatFields set to N produces the following output in the Teradata PT private log only if all “dump” attributes are set to N:

**Note:**

The output is identical to the output DumpFieldsNoTrans produces.

Record Number : 20
COL001        34 31 31 32 2D 30 32 2D 31 32
COL003        97
COL005        54
COL006        55
COL007        56
COL009        FF FE F5 CC
COL012        80 00 8C C0
COL015        00 01 00 30 20.

The output shows:

- The record number
- The column name
- The field itself in hexadecimal characters from the beginning of the field.

# SQL Inserter Operator

## SQL Inserter Operator Capabilities

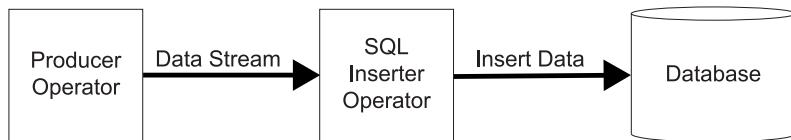
The SQL Inserter operator is a consumer operator inserts the data into a database table. An SQL Inserter operation is similar in function to a BTEQ import operation.

The SQL Inserter operator supports multiple instances and each instance can log one or more SQL sessions. Multiple, parallel instances can be used to improve the performance of loading data records into a target database table. The target table can be an empty table or a table with existing data in it.

### Note:

- If multiple sessions are used and the data has duplicate primary index values, this could result in a potential database deadlock 2631 error. To avoid this error, do not use multiple sessions with the TPT SQL Inserter operator.
- If multiple sessions are used and the data accesses the same row-level lock on the database, the job could result in a hang. To avoid a hang, do not use multiple sessions with the TPT SQL Inserter operator. Use only one session.

The following figure shows the SQL Inserter operator interface.



## The SQL Inserter Operator Function in a Teradata Job Script

When you use the SQL Inserter operator in a Teradata PT job, it does the following:

1. Log on to the database, using your user name, password, database name, and account ID information specified in the job script.
2. Load the data from the Teradata PT data stream into a target table specified in the SQL INSERT statement defined in the Teradata PT APPLY statement.

Only one INSERT statement is allowed to be specified in the APPLY statement. If more than one INSERT statement is found in the APPLY statement, the SQL Inserter operator will issue an error message to both console and private logs and terminate the job.

3. Log off the database.

4. If the SQL Inserter operator job is successful, terminate the job and report the total number of records successfully sent to the database.
5. If the job is unsuccessful, terminate the job and provide information about the job so that you can correct any problem and restart the job.

## SQL Inserter Operator and the Load Operator

To load large amounts of data, it is generally better to use the Load operator, but for smaller load jobs, the SQL Inserter might perform better than most load jobs because it does not need to set up multiple sessions to run.

Also, the SQL Inserter operator does not require an active load job. It simply uses standard SQL protocol on a single session. If it is difficult to acquire database resources for running concurrent tasks, the SQL Inserter operator has advantages over the Load operator.

## Syntax

Use the following specification and attributes to define the SQL Inserter operator in a Teradata PT job script.

## Required Specifications

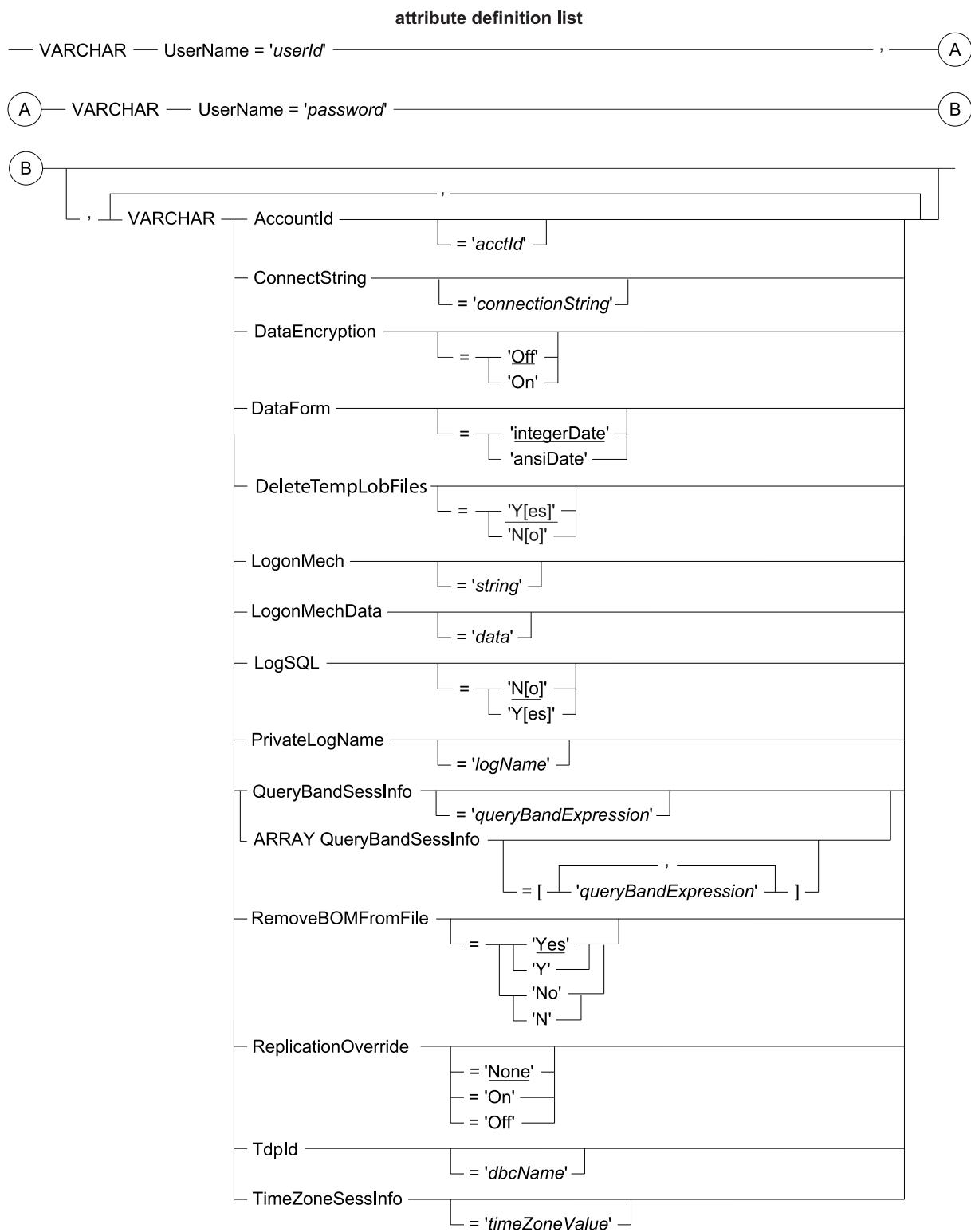
The following specifications are required to define a SQL Inserter operator job.

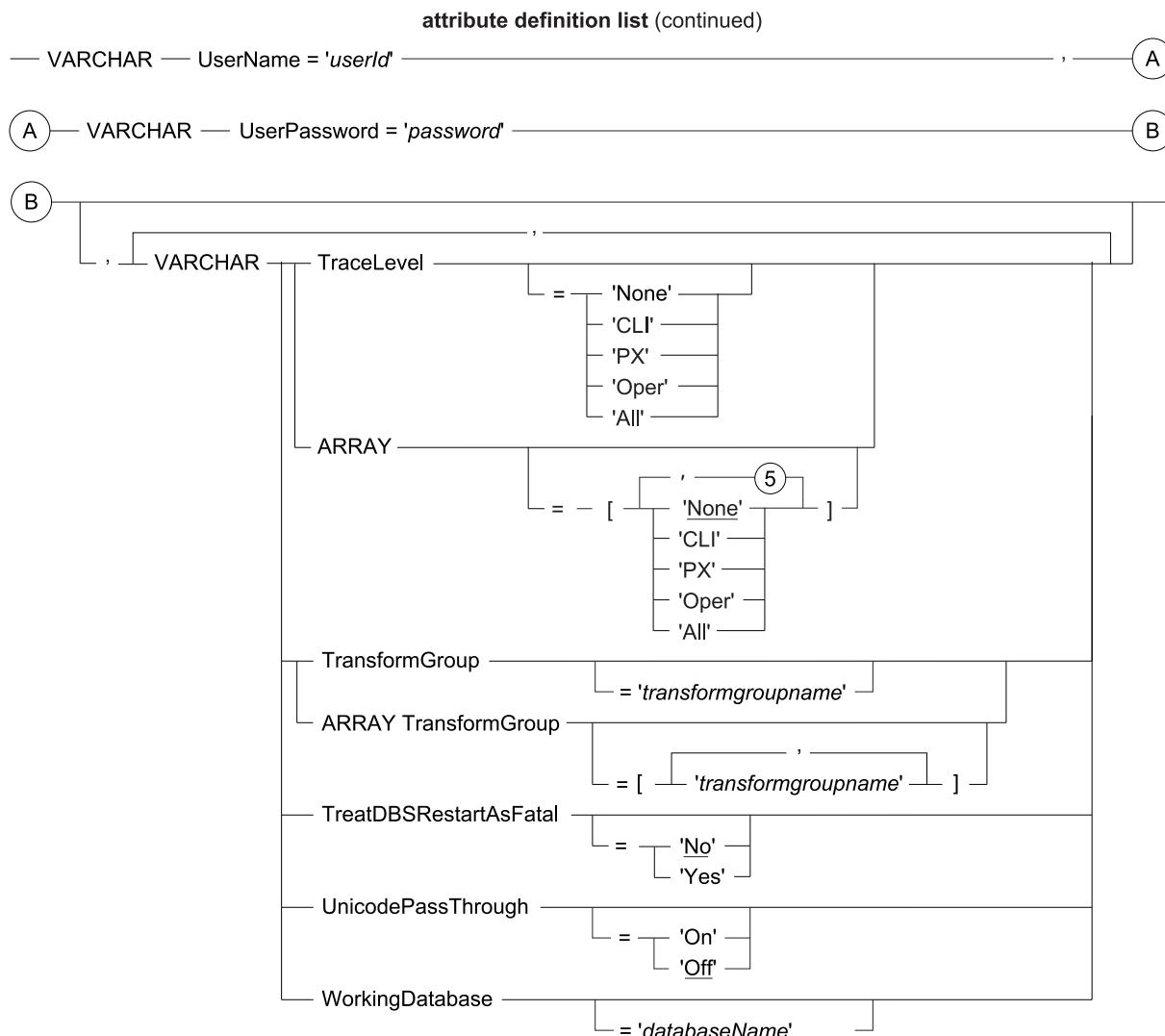
**Required Syntax for the SQL Inserter Operator**

Specification	Description
TYPE	Operator type. Always INSERTER for the SQL Inserter operator.
UserName	Operator attribute providing the name of the user under which the INSERT statement is submitted.
UserPassword	Operator attribute providing the password associated with the user name.

## Required and Optional Attributes

Use the attribute definition list syntax in the Teradata PT DEFINE OPERATOR statement to declare the required attribute values for the SQL Inserter operator.





where:

#### SQL Inserter Operator Attribute Descriptions

Syntax Element	Description
AccountId = 'acctId'	Optional attribute that specifies the account associated with the specified user name. If omitted, it defaults to the account identifier of the immediate owner database.
ARRAY	Optional keyword that specifies more than one attribute value.
ConnectionString = 'connectionString'	Optional attribute that specifies the connection string. The connection string will be passed to CLI. CLI will validate the connection string. For information on connection string, see <i>Teradata® Call-Level Interface Version 2 Reference for Workstation-Attached Systems</i> , B035-2418.

Syntax Element	Description
	<p><b>Note:</b> The TPT Connection String feature is available on all platforms, except for the TDP variant of TPT on z/OS.</p>
DataEncryption = ' <i>option</i> '	<p>Optional attribute that enables full security encryption of SQL requests, responses, and data. Valid values are:</p> <ul style="list-style-type: none"> <li>• 'On' = all SQL requests, responses, and data are encrypted.</li> <li>• 'Off' = no encryption occurs (default).</li> </ul>
DateForm = ' <i>option</i> '	<p>Optional attribute that specifies the DATE data type for the SQL Inserter operator job where:</p> <ul style="list-style-type: none"> <li>• 'integerDate' = the integer DATE data type (default)</li> <li>• 'ansiDate' = the ANSI fixed-length CHAR(10) DATE data type</li> </ul>
DeleteLobDataFiles = ' <i>option</i> '	<p>Optional attribute that specifies whether to delete deferred LOB data files from the Data Connector Producer once the rows are committed to database. Valid values are:</p> <ul style="list-style-type: none"> <li>• 'Yes' (or 'Y') = delete deferred mode LOB data files once rows are committed to the database.</li> <li>• 'No' (or 'N') = do not delete deferred mode LOB data files (default). Specifying any other value results in an error.</li> </ul>
DeleteTempLobFiles	<p>Optional attribute that tells the operator whether or not to delete the temporary LOB directory and the temporary LOB files in the directory at cleanup. Valid values are:</p> <ul style="list-style-type: none"> <li>• 'Y[es]' = Tells the operator to delete the temporary LOB directory at cleanup. This is the default.</li> <li>• 'N[o]' = Tells the operator not to delete the temporary LOB directory at cleanup. The user is responsible for deleting the temporary LOB directory.</li> </ul> <p><b>Note:</b> When the temporary LOB files exist in a remote NFS mount directory, the suggestion is to set this attribute to 'No' and the operator will not spend time deleting the temporary LOB files.</p> <p><b>Note:</b> Temporary LOB files will be created in the temporary LOB directory. The naming convention for the temporary LOB directory will be as follows:</p> <ul style="list-style-type: none"> <li>• When the producer's LobDirectoryPath attribute is set, the name of the temporary LOB directory will be this: &lt;LobDirectoryPath&gt;/&lt;job id&gt;_s&lt;job step number&gt;_TempLOBDir</li> <li>• When the producer's LobDirectoryPath attribute is not set, the name of the temporary LOB directory will be this: &lt;current working dir&gt;/&lt;job id&gt;_s&lt;job step number&gt;_TempLOBDir</li> </ul> <p>This attribute is not supported on z/OS.</p>
LogonMech = ' <i>string</i> '	Optional attribute that specifies which logon mechanism to use.

Syntax Element	Description
	<p><b>Note:</b></p> <p>Specification of this attribute may be required for some authentication methods.</p> <p>The job terminates if the attribute exceeds 8 bytes.</p> <p>For information on specification requirements for LogonMech “Logon Security” in <i>Teradata® Parallel Transporter User Guide</i>, B035-2445.</p>
LogonMechData = ' <i>data</i> '	<p>Optional attribute that passes along additional logon data.</p> <p><b>Note:</b></p> <p>Specification of this attribute is required for some external authentication methods.</p> <p>For information on specification requirements for LogonMechData “Logon Security” <i>Teradata® Parallel Transporter User Guide</i>, B035-2445.</p>
LogSQL = ' <i>option</i> '	<p>Optional attribute that controls how much of the job's SQL to enter into the log.</p> <p>Valid options are:</p> <ul style="list-style-type: none"> <li>• 'Yes' = output the full SQL to the log. The maximum length is 1M.</li> <li>• 'No' = do not output SQL to the log.</li> <li>• No value or attribute omitted = accept the pre-defined limit, which displays up to 32K of SQL if all the SQL is less than 32K. If the SQL to be logged exceeds 32K, truncate the display to the first 32K bytes.</li> </ul>
MaxSessions = ' <i>maxSessions</i> '	<p>Optional attribute that specifies the maximum number of sessions to log on. The MaxSessions value must be greater than 0. Specifying a value less than 1 terminates the job.</p> <p>The default value is one session for each operator instance.</p> <p>The main instance calculates an even distribution of the Inserter operator sessions among the number of instances. For example, if there are 4 instances and 16 sessions, each instance will log on 4 sessions.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>• If multiple sessions are used and the data has duplicate primary index values, this could result in a potential deadlock 2631 error. To avoid this error, do not use multiple sessions with the TPT SQL Inserter operator.</li> <li>• If multiple sessions are used and the data accesses the same row-level lock on the database, the job can result in a hang. To avoid a hang, do not use multiple sessions with the TPT SQL Inserter operator. Use only one session.</li> </ul>
MinSessions = ' <i>minSessions</i> '	<p>Optional attribute that specifies the minimum number of sessions required for the Stream operator job to continue.</p> <p>The MinSessions value must be greater than 0 and less than or equal to the maximum number of sessions defined in the attribute MaxSessions.</p> <p>Specifying a value less than 1 terminates the job.</p>

Syntax Element	Description
PrivateLogName = ' <i>logName</i> '	<p>Optional attribute that specifies the name of a log that is maintained by the Teradata PT Logger inside the public log. The private log contains all the output provided by the operator.</p> <p>The private log can be viewed using the tlogview command as follows, where <i>jobId</i> is the Teradata PT job name and <i>privateLogName</i> is the value for the operator's PrivateLogName attribute:</p> <pre>tlogview -j jobId -f privateLogName</pre> <p>If the private log is not specified, all output is stored in the public log. For more information about the tlogview command, see <a href="#">Teradata PT Utility Commands</a>.</p>
QueryBandSessInfo = ' <i>queryBandExpression</i> '	<p>Optional attribute that specifies the Query Band for the duration of the job sessions.</p> <p>The <i>queryBandExpression</i> is a set of name=value pairs, separated by a semicolon and ending with a semicolon. The user defines the Query Band expression, which will look similar to the following example:</p> <pre>'org=Finance;load=daily;location=west;'</pre> <p>QueryBandSessInfo may also be specified as an ARRAY attribute. For information on the rules for creating a Query Band expression, see <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i>, B035-1144 and <i>Teradata Vantage™ - SQL Data Definition Language Detailed Topics</i>, B035-1184.</p> <p>The value of the QueryBandSessInfo attribute is displayed in the SQL Inserter operator private log.</p> <p>Use of the QueryBandSessInfo attribute is subject to the following rules:</p> <ul style="list-style-type: none"> <li>• By default, Query Band is off until a valid value appears for the QueryBandSessInfo attribute.</li> <li>• If the QueryBandSessInfo attribute contains a value, the SQL Inserter operator constructs the necessary SET QUERY BAND SQL to communicate the request to the database.</li> <li>• The SQL Inserter operator does not check the Query Band expression, but passes the expression to the database as is.</li> <li>• If the version of the database against which the job is being run does not support the Query Band feature, no Query Banding will take place. However, the operator will ignore the error and run the rest of the job.</li> <li>• If there is a syntax error in the Query Band expression, the database will return an error. The SQL Inserter operator will then terminate the job and report the error to the user.</li> </ul>
ReplicationOverride = ' <i>option</i> '	<p>Optional attribute that overrides the normal replication services controls for an active session.</p> <p>Valid values:</p> <ul style="list-style-type: none"> <li>• 'On' = override normal replication services controls for the active session.</li> <li>• 'Off' = override of normal replication services is turned off for the active session (when change data capture is active).</li> <li>• 'None' = (Default) No override request is sent to the database.</li> </ul>

Syntax Element	Description
	<p>For more information, see <i>Teradata Replication Services Using Oracle GoldenGate</i> (B035-1152).</p> <p><b>Note:</b> The user ID that is logged in by the operator must have the REPLCONTROL privilege when setting the value for this attribute.</p>
RemoveBOMFromFile = ' <i>option</i> '	<p>Optional attribute that specifies whether to look for and remove the UTF byte-order-mark (BOM) from the beginning of an XML, JSON, or CLOB data file</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>• 'Y' or 'Yes' = The Inserter operator looks for the UTF BOM at the beginning of an XML, JSON, or CLOB data file and, if found, removes it before processing (default)</li> <li>If the found BOM is not valid for the client Unicode character set, the operator displays an error message and terminates the job.</li> <li>If the BOM is not found, the Inserter operator processes the file without an error message or warning message being logged to the private log or to the console</li> <li>• 'N' or 'No' = The Inserter operator does not look for the UTF BOM. Teradata recommends that RemoveBOMFromFile be set to 'No' when moving XML, JSON or CLOB data in deferred mode encoding in the Unicode character set from one table to another. This can improve loading performance when large amount of data (for example, million rows) are inserted in a table in the deferred mode.</li> </ul> <p>RemoveBOMFromFile can only be used when the following two conditions are met:</p> <ul style="list-style-type: none"> <li>• The job schema has one or more XML, JSON and/or CLOB columns defined AS DEFERRED BY NAME.</li> <li>• The client character set for the load job is the Unicode character set.</li> </ul> <p>If these two conditions are not met, the values specified in the attribute are ignored.</p>
RoleName = ' <i>role name</i> '	<p>Optional attribute that implements security in a database environment. The operator will prepend the value with "SET ROLE ". The syntax will be sent to the database as follows:</p> <pre>SET ROLE &lt;i&gt;role name&lt;/i&gt;;</pre> <p>For example:</p> <pre>SET ROLE All;</pre> <p>For details of "SET ROLE" command use, see <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i>, B035-1144.</p> <p>The VARCHAR ARRAY can specify more than one value, for example:</p> <pre>VARCHAR ARRAY RoleName = [ 'role name1', 'role name2' ],</pre> <p>The operator will send the request to the database on the SQL sessions after the sessions are connected.</p>

Syntax Element	Description
	<p><b>Note:</b></p> <ul style="list-style-type: none"> <li>The operator does not validate the value for this attribute. The operator passes the value to the database as is. The database will validate the value. The operator will terminate the job with an error when the validation fails.</li> <li>C-style comments are allowed in the value and will be passed to the database.</li> <li>ANSI-style comments are not supported in the value. The operator can terminate the job with a syntax error when the value contains an ANSI-style comment.</li> <li>A semicolon is not allowed in the value, because the operator allows only a single statement per request. The operator will terminate the job with an operator error when the value contains a semicolon.</li> </ul>
Tdpld = ' <i>dbcName</i> '	<p>Optional attribute that specifies the name of the database machine (non-mainframe platforms) or TDP (mainframe platforms) for the insert operation. The <i>dbcName</i> can be up to 256 characters and can be a domain server name.</p> <p>If you do not specify the value for the Tdpld attribute, the operator uses the default Tdpld established for the user by the system administrator.</p>
TimeZoneSessInfo = ' <i>timeZoneValue</i> '	<p>Optional attribute that allows you to change the default time zone displacement for the duration of the operator's job session.</p> <p>When you provide a value for this attribute, the operator will build the SET TIME ZONE &lt;<i>timeZoneValue</i>&gt;; SQL request.</p> <p>The operator will send the request to the database on the SQL sessions after the sessions are connected.</p> <p>Here are some examples:</p> <ul style="list-style-type: none"> <li><b>Example 1:</b> This example sets the session default time zone displacement to LOCAL, which is the system default time zone: VARCHAR TimeZoneSessInfo = 'LOCAL'</li> <li><b>Example 2:</b> This example sets the session default time zone displacement to USER, which is the default time zone for the logged on user: VARCHAR TimeZoneSessInfo = 'USER'</li> <li><b>Example 3:</b> This example sets the session default time zone displacement to a simple constant time zone string expression: VARCHAR TimeZoneSessInfo = '''America Pacific'''</li> </ul> <p><b>Note:</b></p> <p>Any single quote character ('') inside the value must be entered as two consecutive single quote characters in a TPT job script. This ensures the correct value will be sent to the database.</p>

Syntax Element	Description
	<p><b>Note:</b></p> <ul style="list-style-type: none"> <li>The operator does not validate the value for this attribute. The operator passes the value to the database as is. The database will validate the value. The operator will terminate the job with an error when the validation fails.</li> <li>C-style comments are allowed in the value and will be passed to the database.</li> <li>ANSI-style comments are not supported in the value. The operator will terminate the job with a syntax error when the value contains an ANSI-style comment.</li> <li>A semicolon is not allowed in the value, because the operator allows only a single statement in the "SET TIME ZONE SQL" request. The operator will terminate the job with an operator error when the value contains a semicolon.</li> </ul> <p>For more information on SET TIME ZONE SQL, see <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i>, B035-1144.</p>
TraceLevel = 'level'	<p>Optional attribute that specifies the types of diagnostic messages that are written by the operator to the public log (or private log, if one is specified using the PrivateLogName attribute). The diagnostic trace function provides more detailed information in the log file to aid in problem tracking and diagnosis.</p> <p>The trace levels are:</p> <ul style="list-style-type: none"> <li>'None' = TraceLevel turned off (default).</li> <li>'CLI' = enables the tracing function for CLI-related activities (interaction with the database)</li> <li>'PX' = enables the tracing function for activities related to the Teradata PT infrastructure</li> <li>'Oper' = enables the tracing function for operator-specific activities</li> <li>'All' = enables tracing for all the mentioned activities</li> </ul> <p>The VARCHAR ARRAY can specify more than one value, for example:</p> <pre>VARCHAR TraceLevel = 'CLI' VARCHAR TraceLevel = 'OPER' VARCHAR ARRAY TraceLevel = [ 'CLI' ] VARCHAR ARRAY TraceLevel = [ 'CLI', 'OPER' ]</pre> <p><b>Note:</b></p> <p>The TraceLevel attribute is provided as a diagnostic aid only. The amount and type of additional information provided by this attribute changes to meet evolving needs from release to release.</p>
TransformGroup = 'transformgroupname'	<p>Optional attribute that supports changing the active transform for Teradata Complex Data Types (CDTs). The value is the <i>&lt;udt name&gt; &lt;transform group name&gt;</i>, and the operator will prepend the hardcoded string "SET TRANSFORM GROUP FOR TYPE ". The syntax sent to RDBMS is as follows:</p> <pre>SET TRANSFORM GROUP FOR TYPE &lt;udt name&gt; &lt;transform group name&gt;;</pre> <p>For example:</p>

Syntax Element	Description
	<pre>"SET TRANSFORM GROUP FOR TYPE JSON CHARACTER SET LATIN TD_ JSON_VARCHAR;"</pre> <p>The VARCHAR ARRAY can specify more than one value, for example:</p> <pre>VARCHAR ARRAY TransformGroup = [ 'JSON CHARACTER SET LATIN TD_ JSON_VARCHAR',                                 'ST_GEOGRAPHY TD_GEO_VARCHAR' ],</pre> <p>The operator will send the request to the database on the SQL sessions after the sessions are connected.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>• The operator does not validate the value for this attribute. The operator passes the value to the database as is. The database will validate the value. The operator will terminate the job with an error when the validation fails.</li> <li>• C-style comments are allowed in the value and will be passed to the database.</li> <li>• ANSI-style comments are not supported in the value. The operator can terminate the job with a syntax error when the value contains an ANSI-style comment.</li> <li>• A semicolon is not allowed in the value, because the operator allows only a single statement per request. The operator will terminate the job with an operator error when the value contains a semicolon.</li> </ul>
TreatDBSRestartAsFatal= 'option'	<p>Optional attribute that tells the operator whether to terminate the job when a database restart occurs.</p> <p>The TreatDBSRestartAsFatal values are:</p> <ul style="list-style-type: none"> <li>• 'No' ('N') = The operator will not terminate if a database restart occurs (default). The database restart will be treated as a retryable one.</li> <li>• 'Yes' ('Y') = The operator will terminate if a database restart occurs.</li> </ul>
UnicodePassThrough = 'value'	<p>Optional attribute that tells the operator to enable or disable the Unicode Pass Through feature.</p> <p>Valid values:</p> <ul style="list-style-type: none"> <li>• 'On' = Enable the Unicode Pass Through feature in the operator.</li> <li>• 'Off' = (Default) Disable the Unicode Pass Through feature in the operator.</li> </ul> <p><b>Note:</b></p> <p>When a TPT job is using the UTF8 or UTF16 session character set, the UnicodePassThrough attribute can be set to 'On' to allow the operator to load data with Unicode pass through characters.</p>
UserPassword = 'password'	<p>Attribute that specifies the password associated with the user name.</p> <p><b>Note:</b></p> <p>Use of this attribute is not compatible with some external authentication logon methods. For more information on password specification requirements, see "Logon Security" in <i>Teradata® Parallel Transporter User Guide</i>, B035-2445.</p>

Syntax Element	Description
VARCHAR	Keyword that specifies VARCHAR as the data type of the defined attribute.
WorkingDatabase = 'databaseName'	<p>Optional attribute that specifies a database other than the logon database as the default database.</p> <p>The name of the database that is specified with this attribute is used in the SQL DATABASE statement that is sent by the operator immediately after connecting the two SQL sessions.</p> <p>If WorkingDatabase is not specified, the default database associated with the logged on user is assumed for all unqualified table names.</p>

## Job Options

### Support for LOBs

For details [Large Objects](#).

### Data Integrity

To protect data integrity, the SQL Inserter operator treats the entire loading job as a single explicit transaction.

If any error is encountered during the insert operation, the SQL Inserter operator backs out all rows of data inserted up to that point. For example, the SQL Inserter operator immediately terminates a load job if a duplicate data row is inserted into a target table.

### Data Loading

The SQL Inserter operator is not used for massive data loading because it is slower than the other load operators.

### Usage Notes

#### Enhanced LOB Support

The SQL Inserter operator supports enhanced LOB functionality.

When the TPT job has one or more LOBs (inline or deferred, regardless of the schema size), the SQL Inserter operator can use up to 7 MB message size for sending data to the database. The database has a limit of 7 MB message size for such jobs.

For more information, see [Enhanced LOB Support](#).

# Operational Considerations

## Checkpointing and Restarting

The SQL Inserter operator takes two basic checkpoints, a start-of-data checkpoint and an end-of-data checkpoint, during a load operation so that the SQL Inserter operator can restart automatically when the load operation is interrupted by a database restart, should one occur.

After an error, the Inserter operator does not have the capability to restart the load job where it left off. Using the interval checkpoint is not recommended for most load jobs using the Inserter operator; however, if a target table is empty and millions of data rows with many JSON, XML, or LOB columns are loaded, the interval checkpoint option could be used to avoid a potential of deadlock errors during the load job.

For information on checkpoints and restarts, see the *Teradata® Parallel Transporter User Guide*, B035-2445.

## Restrictions and Limitations

This section describes the restrictions and limitations when using the SQL Inserter operator.

### Data Types

- The Inserter operator can load CLOB/BLOB/JSON/XML in inline and deferred modes on all non-z/OS platforms.
  - On z/OS, the operator can load CLOB/BLOB/JSON/XML in inline mode if the size is less than or equal to 64,000 bytes. It cannot load CLOB/BLOB/JSON/XML in deferred mode.
- TPT does not support the ST\_Geometry data type in the TPT schema. But, TPT *can* load data to ST\_Geometry columns if the job specifies CLOB or BLOB in the TPT schema.
- UDT data can be loaded in its external type format.

### Error Tables

The SQL Inserter operator does not have error tables like the Load, Update, and Stream operators.

### Other Notes

- One SQL Inserter operator job can load a single database table or view.
- The target table must already exist.
- Data can be loaded into an empty or populated table.
- Only the Teradata SQL INSERT statement is supported. Any other Teradata SQL statements are not supported.

- One Teradata SQL INSERT statement is allowed for a single SQL Inserter operator job. Multiple statements are not allowed.
- One DML group is allowed for a single SQL Inserter operator job. Multiple DML groups are not allowed.
- The SQL Inserter operator treats the entire loading job as a single explicit transaction. If any error is encountered during the insert operation, the SQL Inserter operator backs out all rows of data inserted up to that point and the job terminates.
- The target table cannot be rolled back after the job is complete with no errors.
- Row hash level locking is used during the job. The job does not lock the target table.
- The SQL Inserter operator can support multiple instances, and each instance can log on one or more sessions based on the valued specified in the attribute MaxSessions.

---

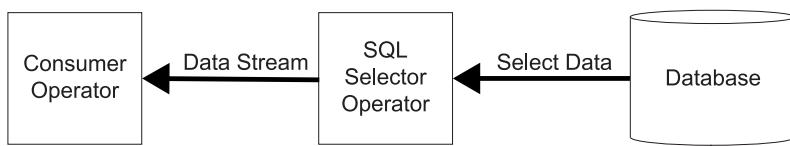
**Note:**

- If multiple sessions are used and the data has duplicate primary index values, this could result in a potential database deadlock 2631 error. To avoid this error, do not use multiple sessions with the TPT SQL Inserter operator.
  - If multiple sessions are used and the data accesses the same row-level lock on the database, the job could result in a hang. To avoid a hang, do not use multiple sessions with the TPT SQL Inserter operator. Use only one session.
- 
- The SQL Inserter operator does not require a database load slot.

# SQL Selector Operator

## SQL Selector Operator Capabilities

The SQL Selector operator is a producer operator that submits Teradata SQL SELECT statements to the database to retrieve data from a table.



## SQL Selector Operator and the Export Operator Compared

The main differences between the SQL Selector operator and the Export operator are seen in their performance and feature sets.

- The Export operator allows multiple sessions and multiple instances to extract data from the database. When exporting a large number of rows, it is usually better to use the Export operator.
- The SQL Selector operator only allows one session and one instance to extract rows from the database. When exporting a small number of rows, the SQL Selector operator usually performs better than the Export operator.

The SQL Selector operator support of a single session and a single instance is similar to a BTEQ Export operation.

## Advantages of the SQL Selector Operator

The SQL Selector operator has features not found in the Export operator such as field mode processing:

- The SQL Selector operator has a Report Mode, known as Field Mode in the BTEQ environment. All data retrieved in this mode is converted to character strings.
- The SQL Selector operator is the only operator that can retrieve data from the database in Field Mode and send data to the DataConnector operator to have it written to an external target in the VARTEXT (or TEXT) format. The Export operator cannot be used to extract data from a table and write it to an external target in VARTEXT (or TEXT) format.
- The SQL Selector operator can extract JSON, XML, or LOB data from the database.
- The SQL Selector operator does not require an active load job. Instead, standard SQL protocol is used on the single session. If it is difficult to acquire database resources for running concurrent tasks, the SQL Selector operator is a logical choice rather than the Export operator.

## Defining an SQL Selector Operator in a Teradata PT Script

The SQL Selector operator definition must provide the following specifications:

- TYPE SQL Selector
- Key operator attributes:
  - TdplId
  - UserName
  - UserPassword
  - SelectStmt
  - PrivateLogName
  - **For LOB exporting:** LobDirectoryPath
  - **For LOB loading:** LobFileName
  - **For LOB loading:** LobFileExtension
  - **For non-LOB loading:** ReportModeOn

## Syntax

Use the following specifications and attributes to define the SQL Selector operator in a Teradata PT job script.

## Required Specifications

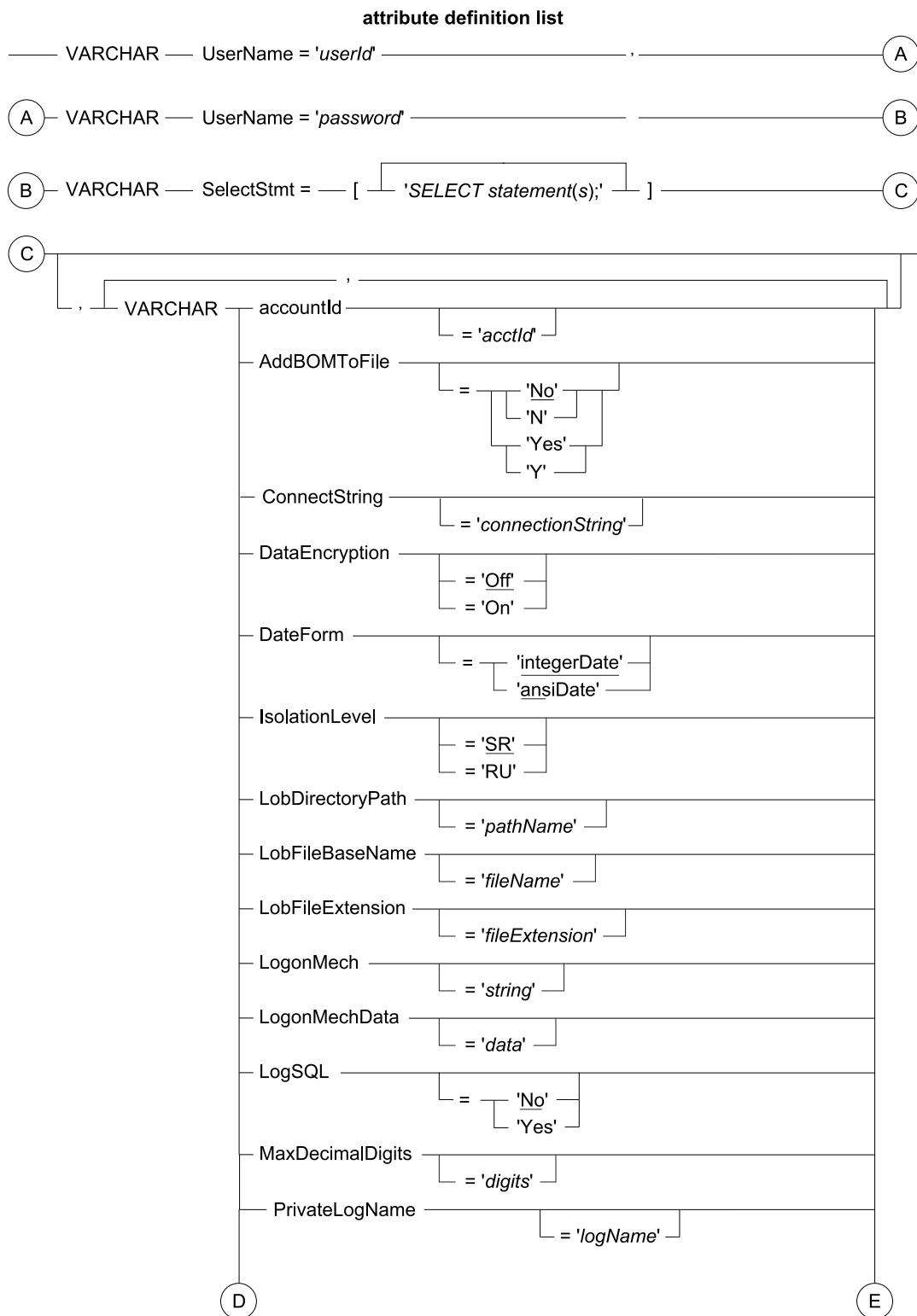
The following specifications are required to define the SQL Selector operator in a Teradata PT job script.

### Required Syntax for the SQL Selector Operator

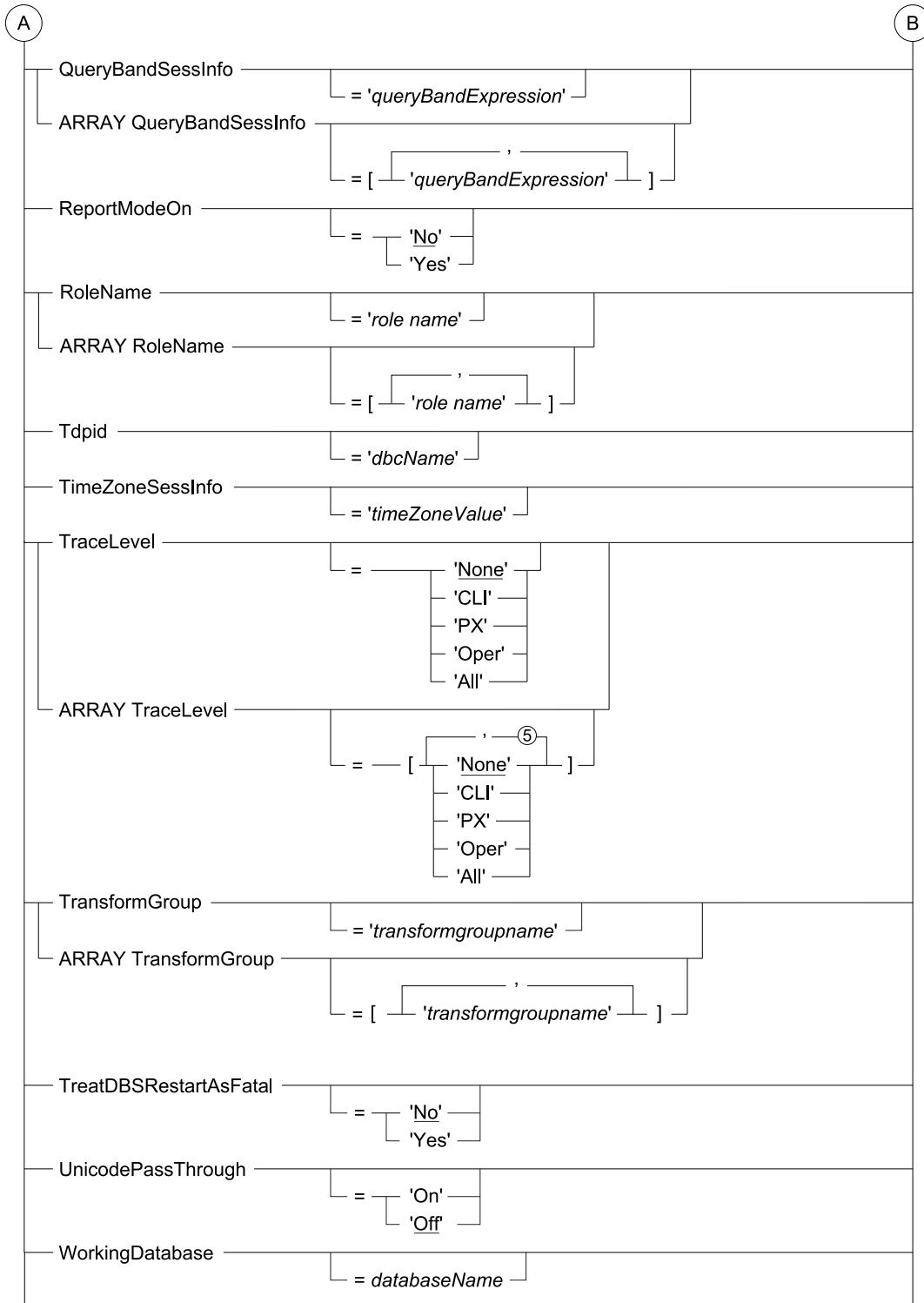
Specification	Description
TYPE	Operator type. Always SELECTOR for the SQL Selector operator.
UserName	Operator attribute providing the name of the user under which the Teradata SQL SELECT statement is submitted.
UserPassword	Operator attribute providing the password associated with the user name.
SelectStmt	Operator attribute providing the Teradata SQL SELECT statement. The SQL Selector operator supports one or more SELECT statements.

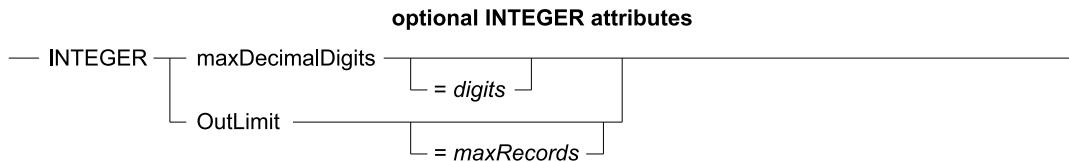
## Required and Optional Attributes

Use the attribute definition list syntax in the Teradata PT DEFINE OPERATOR statement to declare the required attribute values for the SQL Selector operator.



## attribute definition list (continued)





where:

Syntax Element	Description
AccountId = 'acctId'	<p>Optional attribute that specifies the account associated with the specified user name.</p> <p>If omitted, it defaults to the account identifier of the immediate owner database.</p>
AddBOMToFile = 'option'	<p>Optional attribute that specifies whether the UTF byte-order-mark (BOM) will be added at the beginning of an XML, JSON, or CLOB output data file.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>'Y' or 'Yes' = The Selector operator adds the appropriate UTF BOM at the beginning of an XML, JSON or CLOB output data file.</li> <li>'N' or 'No' = The Selector operator does not prefix the appropriate UTF Byte-Order-Mark (BOM) at the beginning of an XML (JSON or CLOB) output data file (default).</li> </ul> <p>AddBOMToFile can only be used when following two conditions are met:</p> <ol style="list-style-type: none"> <li>1. The job schema has one or more XML, JSON, or CLOB column(s) defined AS DEFERRED BY NAME.</li> <li>2. The client character set for the load job is the Unicode character set.</li> </ol> <p>If these two conditions are not met, the values specified in the attribute are ignored.</p> <p><b>Note:</b></p> <p>The operator does not add a BOM to a data file extracted from a BLOB column in deferred mode.</p> <p>Because the default behavior is not to prefix a BOM when extracting data from XML, JSON, or CLOB columns, you should set the attribute RemoveBOMFromFile to 'No' if you use the Selector to extract XML, JSON, or CLOB data and the Inserter operator to load the data in deferred mode. For more information about RemoveBOMFromFile attribute, see <a href="#">SQL Inserter Operator</a>.</p> <p><b>Note:</b></p> <p>Teradata strongly recommends that you specify XMLSERIALIZE on selected XML columns so that the byte-order-mark (BOM) matches the XML encoding when using the client UTF-16 character set.</p>
ARRAY	Optional keyword that specifies more than one attribute value.
ConnectionString = 'connectionString'	<p>Optional attribute that specifies the connection string. The connection string will be passed to CLI. CLI will validate the connection string.</p> <p>For information on connection string, see <i>Teradata® Call-Level Interface Version 2 Reference for Workstation-Attached Systems</i>, B035-2418.</p>

Syntax Element	Description
	<p><b>Note:</b> The TPT Connection String feature is available on all platforms, except on z/OS.</p>
DataEncryption = ' <i>option</i> '	<p>Optional attribute that enables full security encryption of SQL requests, responses, and data. Valid values are:</p> <ul style="list-style-type: none"> <li>• 'On' = all SQL requests, responses, and data are encrypted.</li> <li>• 'Off' = no encryption occurs (default).</li> </ul>
DateForm = ' <i>option</i> '	<p>Optional attribute that specifies the DATE data type for the SQL Selector operator job. The values are:</p> <ul style="list-style-type: none"> <li>• 'integerDate' = the integer DATE data type (default)</li> <li>• 'ansiDate' = the ANSI fixed-length CHAR(10) DATE data type</li> </ul>
INTEGER	Keyword that specifies INTEGER as the data type of the defined attribute.
IsolationLevel = ' <i>option</i> '	<p>Optional attribute that specifies a lock access value that determines the isolation level for all queries in a session. Valid values:</p> <ul style="list-style-type: none"> <li>• RU = read uncommitted</li> <li>• SR = (default) serializable</li> </ul> <p>When the database successfully changes the isolation level, the following message is written to the private log:</p> <pre>Session Isolation Level: RU</pre>
LobDirectoryPath = ' <i>pathName</i> '	Optional attribute that specifies the complete path name of an existing directory where all LOB, JSON, and XML data files will be written.
LobFileNameBase = ' <i>fileName</i> '	<p>Optional attribute that defines a character string that will be prefixed to the names of LOB, JSON, and XML data files. The file names created by the SQL Selector operator are in the following format:</p> <pre>&lt;column-name&gt;_c&lt;#&gt;_&lt;job-id&gt;_p&lt;#&gt;_r&lt;#&gt;</pre> <p>where:</p> <ul style="list-style-type: none"> <li>• # that follows "c" is the column number.</li> <li>• # that follows "p" is the identification of the SQL Selector copy.</li> <li>• # that follows "r" is the row order returned from the database.</li> </ul> <p><b>Note:</b> The column names are sanitized to replace any character that is considered invalid for a Windows file name with an underscore '_'. These are the 9 characters that get replaced: \ / : * ? " &lt; &gt;   For consistency, the substitutions are done for UNIX platforms too. The '_c&lt;#&gt;' is added to ensure uniqueness of generated file names.</p> <p>For example, if we have the following schema:</p>

Syntax Element	Description
	<pre>DEFINE SCHEMA &lt;schema-name&gt; (     COL2 BLOB AS DEFERRED BY NAME,     COL3 CLOB AS DEFERRED BY NAME,     COL4 XML AS DEFERRED BY NAME,     COL5 JSON(1000000) AS DEFERRED BY NAME, );</pre> <p>where LobFileBaseName has the value "my_test", then the files names will be:</p> <ul style="list-style-type: none"> <li>• my_test_COL2_c1_&lt;job-id&gt;_p1_r1</li> <li>• my_test_COL3_c2_&lt;job-id&gt;_p1_r1</li> <li>• my_text_COL4_c3_&lt;job-id&gt;_pl_r1</li> <li>• my_text_COL5_c4_&lt;job-id&gt;_pl_r1</li> <li>• my_test_COL2_c1_&lt;job-id&gt;_p1_r2</li> <li>• my_test_COL3_c2_&lt;job-id&gt;_p1_r2</li> <li>• my_test_COL4_c3_&lt;job-id&gt;_pl_r2</li> <li>• my_test_COL5_c4_&lt;job-id&gt;_pl_r2</li> <li>• my_test_COL2_c1_&lt;job-id&gt;_p1_r3</li> <li>• my_test_COL3_c2_&lt;job-id&gt;_p1_r3</li> <li>• my_test_COL4_c3_&lt;job-id&gt;_pl_r3</li> <li>• my_test_COL5_c4_&lt;job-id&gt;_pl_r3</li> </ul> <p>and so on.</p>
LobFileExtension = ' <i>fileExtension</i> '	<p>Optional attribute that specifies the extension for LOB, JSON, and XML data file names.</p> <p>Examples of 'file-extensions' include:</p> <ul style="list-style-type: none"> <li>• 'jpg': indicates that a file 'ccc.jpg' is a picture file.</li> <li>• 'gif': indicates that the file 'ddd.gif' is a picture file.</li> <li>• 'html': indicates that the file 'aaa.html' is an html file.</li> <li>• * 'json': indicates that the file is a JSON file.</li> </ul>
LogonMech = ' <i>string</i> '	<p>Optional attribute that specifies which logon mechanism to use.</p> <p><b>Note:</b></p> <p>Specification of this attribute may be required for some authentication methods.</p> <p>The job terminates if the attribute exceeds 8 bytes.</p> <p>For information on specification requirements for LogonMech, see "Logon Security" in <i>Teradata® Parallel Transporter User Guide</i>, B035-2445.</p>
LogonMechData = ' <i>data</i> '	<p>Optional attribute that passes along additional logon data.</p> <p><b>Note:</b></p> <p>Specification of this attribute is required for some external authentication methods. For information on specification requirements for LogonMechData, see "Logon Security" in <i>Teradata® Parallel Transporter User Guide</i>, B035-2445.</p>

Syntax Element	Description
LogSQL = ' <i>option</i> '	<p>Optional attribute that controls how much of the job's SQL to enter into the log.</p> <p>Valid options are:</p> <ul style="list-style-type: none"> <li>'Yes' = output the full SQL to the log. The maximum length is 1M.</li> <li>'No' = do not output SQL to the log.</li> <li>No value or attribute omitted = accept the pre-defined limit, which displays up to 32K of SQL if all the SQL is less than 32 K. If the SQL to be logged exceeds 32K, truncate the display to the first 32 K bytes.</li> </ul>
MaxDecimalDigits = <i>maxDecimalDigits</i>	<p>Optional attribute that specifies the maximum number of digits in the DECIMAL data type that can be exported.</p> <p>The default value is 38.</p> <p>If the attribute has a value of <i>q</i> where <math>38 \geq q \geq 1</math>, then any returned DECIMAL (<i>n,[m]</i>) data item with <i>n&gt;q</i> is implicitly CAST to DECIMAL; overflows are handled the same as an explicit CAST.</p> <p>On mainframe-attached platforms, it is recommended that the MaxDecimalDigits attribute not exceed 31 to avoid representations that exceed the capacity of the native instruction set.</p>
OutLimit= <i>maxRecords</i>	<p>Optional attribute that specifies the maximum number of records processed by each instance of the operator.</p> <pre>INTEGER OutLimit = 1000</pre> <p>If specified, the OutLimit value must be greater than 0.</p> <p>If OutLimit is not specified, the number of records processed is not limited.</p>
PrivateLogName = ' <i>logName</i> '	<p>Optional attribute that specifies the name of a log that is maintained by the Teradata PT Logger inside the public log. The private log contains all the output provided by the operator.</p> <p>The private log can be viewed using the tlogview command as follows, where <i>jobId</i> is the Teradata PT job name and <i>privateLogName</i> is the value for the operator PrivateLogName attribute:</p> <pre>tlogview -j jobId -f privateLogName</pre> <p>If the private log is not specified, all the output is stored in the public log.</p> <p>For more information about the tlogview command, see <a href="#">Teradata PT Utility Commands</a>.</p>
QueryBandSessInfo = ' <i>queryBandExpression</i> '	<p>Optional attribute that specifies the Query Band for the duration of the job sessions.</p> <p>The <i>queryBandExpression</i> is a set of name=value pairs, separated by a semicolon and ending with a semicolon. The user defines the Query Band expression, which will look similar to the following example:</p> <pre>'org=Finance;load=daily;location=west;'</pre> <p>QueryBandSessInfo may also be specified as an ARRAY attribute.</p> <p>For information on the rules for creating a Query Band expression, see <a href="#">Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</a>, B035-1144 and <a href="#">Teradata Vantage™ - SQL Data Definition Language Detailed Topics</a>, B035-1184.</p>

Syntax Element	Description
	<p>The value of the QueryBandSessInfo attribute is displayed in the SQL Selector operator private log.</p> <p>Use of the QueryBandSessInfo attribute is subject to the following rules:</p> <ul style="list-style-type: none"> <li>• By default, Query Band is off until a valid value appears for the QueryBandSessInfo attribute.</li> <li>• If the QueryBandSessInfo attribute contains a value, the SQL Selector operator constructs the necessary SET QUERY BAND SQL to communicate the request to the database.</li> <li>• The SQL Selector operator does not check the Query Band expression, but passes the expression to the database as is.</li> <li>• If the version of the database against which the job is being run does not support the Query Band feature, no Query Banding will take place. However, the operator will ignore the error and run the rest of the job.</li> <li>• If there is a syntax error in the Query Band expression, the database will return an error. The SQL Selector operator will then terminate the job and report the error to the user.</li> </ul>
ReportModeOn = ' <i>option</i> '	<p>Optional attribute that specifies whether to use the field report mode. This feature allows you to extract data from the database in character form, and then change it into the “delimited” (VARTEXT) format using the DataConnector operator and save it to a text file. When the data is exported in character format (field report mode), columns defined in the schema must be VARCHAR.</p> <p>The values are:</p> <ul style="list-style-type: none"> <li>• 'Yes' ('Y') = Report mode (returns data in character format)</li> <li>• 'No' ('N') = Indicator mode (default)</li> </ul>
RoleName = ' <i>role name</i> '	<p>Optional attribute that implements security in a database environment. The operator will prepend the value with "SET ROLE ". The syntax will be sent to the database as follows:</p> <pre>SET ROLE &lt;role name&gt;;</pre> <p>For example:</p> <pre>SET ROLE All;</pre> <p>For details of "SET ROLE" command use, see <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i>, B035-1144.</p> <p>The VARCHAR ARRAY can specify more than one value, for example:</p> <pre>VARCHAR ARRAY RoleName = [ 'role name1', 'role name2' ],</pre> <p>The operator will send the request to the database on the SQL session after the session is connected.</p>

Syntax Element	Description
	<p><b>Note:</b></p> <ul style="list-style-type: none"> <li>The operator does not validate the value for this attribute. The operator passes the value to the database as is. The database will validate the value. The operator will terminate the job with an error when the validation fails.</li> <li>C-style comments are allowed in the value and will be passed to the database.</li> <li>ANSI-style comments are not supported in the value. The operator can terminate the job with a syntax error when the value contains an ANSI-style comment.</li> <li>A semicolon is not allowed in the value, because the operator allows only a single statement per request. The operator will terminate the job with an operator error when the value contains a semicolon.</li> </ul>
SelectStmt = 'SELECT <i>statements</i> ;'	<p>Required attribute that specifies a Teradata SQL SELECT statement or statements that return row data in the form of a result table.</p> <p>The SQL Selector operator can submit a single SELECT statement or multiple SELECT statements. All specified SELECT statements are treated as a single request to be sent to the database. The entire request cannot be larger than 1 MB (Vantage limitation).</p>
Tdpld = ' <i>dbcName</i> '	<p>Optional attribute that specifies the name of the database machine (non-mainframe platforms) or TDP (mainframe platforms) for the SQL SELECT statement.</p> <p>The <i>dbcName</i> can be up to 256 characters and can be a domain server name. If you do not specify value for the Tdpld attribute, the operator uses the default TdplID established for the user by the system administrator.</p>
TimeZoneSessInfo = ' <i>timeZoneValue</i> '	<p>Optional attribute that allows you to change the default time zone displacement for the duration of the operator's job session.</p> <p>When you provide a value for this attribute, the operator will build the SET TIME ZONE &lt;<i>timeZoneValue</i>&gt;; SQL request.</p> <p>The operator will send the request to the database on the SQL session after the session is connected.</p> <p>Here are some examples:</p> <ul style="list-style-type: none"> <li><b>Example 1:</b> This example sets the session default time zone displacement to LOCAL, which is the system default time zone: VARCHAR TimeZoneSessInfo = 'LOCAL'</li> <li><b>Example 2:</b> This example sets the session default time zone displacement to USER, which is the default time zone for the logged on user: VARCHAR TimeZoneSessInfo = 'USER'</li> <li><b>Example 3:</b> This example sets the session default time zone displacement to a simple constant time zone string expression: VARCHAR TimeZoneSessInfo = '''America Pacific'''</li> </ul> <p><b>Note:</b></p> <p>Any single quote character ('') inside the value must be entered as two consecutive single quote characters in a TPT job script. This ensures the correct value will be sent to the database.</p>

Syntax Element	Description
	<p><b>Note:</b></p> <ul style="list-style-type: none"> <li>The operator does not validate the value for this attribute. The operator passes the value to the database as is. The database will validate the value. The operator will terminate the job with an error when the validation fails.</li> <li>C-style comments are allowed in the value and will be passed to the database.</li> <li>ANSI-style comments are not supported in the value. The operator will terminate the job with a syntax error when the value contains an ANSI-style comment.</li> <li>A semicolon is not allowed in the value, because the operator allows only a single statement in the "SET TIME ZONE SQL" request. The operator will terminate the job with an operator error when the value contains a semicolon.</li> </ul> <p>For more information on SET TIME ZONE SQL, see <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i>, B035-1144.</p>
TraceLevel = 'level'	<p>Optional attribute that specifies the types of diagnostic messages that are written by each instance of the operator to the public log (or private log, if one is specified using the PrivateLogName attribute). The diagnostic trace function provides more detailed information in the log file to aid in problem tracking and diagnosis.</p> <p>The trace levels are:</p> <ul style="list-style-type: none"> <li>'None' = TraceLevel turned off (default).</li> <li>'CLI' = enables the tracing function for CLI-related activities (interaction with the database)</li> <li>'PX' = enables the tracing function for activities related to the Teradata PT infrastructure</li> <li>'Oper' = enables the tracing function for operator-specific activities</li> <li>'All' = enables tracing for all the mentioned activities</li> </ul> <p>The VARCHAR ARRAY can specify more than one value, for example:</p> <pre>VARCHAR TraceLevel = 'CLI' VARCHAR TraceLevel = 'OPER' VARCHAR ARRAY TraceLevel = [ 'CLI' ] VARCHAR ARRAY TraceLevel = [ 'CLI', 'OPER' ]</pre> <p><b>Note:</b></p> <p>The TraceLevel attribute is provided as a diagnostic aid only. The amount and type of additional information provided by this attribute will change to meet evolving needs from release to release.</p>
TransformGroup = 'transformgroupname'	<p>Optional attribute that supports changing the active transform for Teradata Complex Data Types (CDTs). The value is the &lt;udt name&gt; &lt;transform group name&gt;, and the operator will prepend the hardcoded string "SET TRANSFORM GROUP FOR TYPE ". The syntax sent to RDBMS is as follows:</p> <pre>SET TRANSFORM GROUP FOR TYPE &lt;udt name&gt; &lt;transform group name&gt;;</pre>

Syntax Element	Description
	<p>For example:</p> <pre>"SET TRANSFORM GROUP FOR TYPE JSON CHARACTER SET LATIN TD_JSON_VARCHAR;"</pre> <p>The VARCHAR ARRAY can specify more than one value, for example:</p> <pre>VARCHAR ARRAY TransformGroup = [ 'JSON CHARACTER SET LATIN TD_JSON_VARCHAR',                                 'ST_Geometry TD_Geo_VARCHAR' ],</pre> <p>The operator will send the request to the database on the SQL session after the session is connected.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>• The operator does not validate the value for this attribute. The operator passes the value to the database as is. The database will validate the value. The operator will terminate the job with an error when the validation fails.</li> <li>• C-style comments are allowed in the value and will be passed to the database.</li> <li>• ANSI-style comments are not supported in the value. The operator can terminate the job with a syntax error when the value contains an ANSI-style comment.</li> <li>• A semicolon is not allowed in the value, because the operator allows only a single statement per request. The operator will terminate the job with an operator error when the value contains a semicolon.</li> </ul>
TreatDBSRestartAsFatal= 'option'	<p>Optional attribute that tells the operator whether to terminate the job when a database restart occurs.</p> <p>The TreatDBSRestartAsFatal values are:</p> <ul style="list-style-type: none"> <li>• 'No' ('N') = The operator will not terminate if a database restart occurs (default). The database restart will be treated as a retryable one.</li> <li>• 'Yes' ('Y') = The operator will terminate if a database restart occurs.</li> </ul>
UnicodePassThrough = 'value'	<p>Optional attribute that tells the operator to enable or disable the Unicode Pass Through feature.</p> <p>Valid values:</p> <ul style="list-style-type: none"> <li>• 'On' = Enable the Unicode Pass Through feature in the operator.</li> <li>• 'Off' = (Default) Disable the Unicode Pass Through feature in the operator.</li> </ul> <p><b>Note:</b></p> <p>When a TPT job is using the UTF8 or UTF16 session character set, the UnicodePassThrough attribute can be set to 'On' to allow the operator to export data with Unicode pass through characters.</p>
UserName = 'userId'	<p>Attribute that specifies the user name.</p> <p><b>Note:</b></p> <p>Use of this attribute is not compatible with some external authentication logon methods. For more information on UserName specification requirements, see "Logon Security" in <i>Teradata® Parallel Transporter User Guide</i>, B035-2445.</p>

Syntax Element	Description
UserPassword = 'password'	<p>Attribute that specifies the password associated with the user name.</p> <p><b>Note:</b> Use of this attribute is not compatible with some external authentication logon methods. For more information on password specification requirements, see “Logon Security” in <i>Teradata® Parallel Transporter User Guide</i>, B035-2445.</p>
VARCHAR	Keyword that specifies VARCHAR as the data type of the defined attribute.
WorkingDatabase = 'databaseName'	<p>Optional attribute that specifies a database other than the logon database as the default database.</p> <p>The name of the database that is specified with this attribute is used in the SQL DATABASE statement that is sent by the operator immediately after connecting the two SQL sessions.</p> <p>If WorkingDatabase is not specified, the default database associated with the logged on user is assumed for all unqualified table names.</p>

## Usage Notes

### SelectStmt

The SQL Selector operator supports a single SQL SELECT statement or multiple SELECT statements.

Certain restrictions apply when using a SQL SELECT statement in the SQL Selector operator:

- Do not specify a WITH clause with a SELECT statement. Use of a WITH or WITH BY clause produces an error message.
- Do not specify a USING modifier with a SELECT statement. Use of a USING modifier produces an error message.

---

#### Note:

The SQL Selector operator only logs on to one database session. Multiple SELECT statements are therefore sent as one request to the database. The database executes these multiple statements sequentially. Thus, the SQL Selector operator cannot take advantage of the parallel processing environment that Teradata PT offers. To take advantage of parallel processing, break multiple SQL SELECT statement requests into single SELECT statements carried by multiple SQL Selector operators, and then unite these data sources together with the UNION clause.

---

## LOB Loading Attributes

The following attributes:

- LobDirectoryPath

- LobFileName
- LobFileExtension

are only for extracting LOB or JSON data in deferred mode. That is, when LOB columns are defined as follows in a schema:

```
BLOB AS DEFERRED BY NAME
CLOB AS DEFERRED BY NAME
JSON(<length>)AS DEFERRED BY NAME
```

## Enhanced LOB Support

The SQL Selector operator supports enhanced LOB functionality.

When the TPT job has one or more LOBs (inline or deferred, regardless of the schema size), the SQL Selector operator can use up to 16 MB message size for exporting data from the database. The database has a limit of 16 MB message size for such jobs.

For more information, see [Enhanced LOB Support](#).

## Restrictions and Limitations

This section describes the restrictions and limitations when using the SQL Selector operator.

### SELECT Statements

Only the Teradata SQL SELECT statements are supported in the SQL Selector operator's SelectStmt attribute.

The SELECT statements in the SQL Selector operator's SelectStmt attribute cannot have any of the following characteristics:

- Contain a USING modifier
- Contain a WITH or WITH BY clause

All SELECT statements in the SQL Selector operator's SelectStmt attribute are treated as a single multi-statement request to be sent to the database.

The entire length for the SELECT statements cannot be greater than 1MB.

### Data Types

- TPT does not support the ST\_Geometry data type in the TPT schema. But, the SQL Selector operator *can* export ST\_Geometry columns if the job specifies CLOB or BLOB in the TPT schema.
- The SQL Selector operator is the only operator that can export LOB/JSON/XML data types from the database.

## Other Notes

- The SQL Selector operator can export rows from multiple tables and views as long as the result set adheres to a single schema.
- The SQL Selector operator is the only operator that can extract rows from the database in field mode. All data retrieved in the field mode is converted to character strings. To take advantage of this feature, ReportMode must be enabled. When ReportMode is enabled, the columns defined in the script schema must be VARCHAR.
- The SQL Selector operator can export rows from tables and views. The maximum size for each row is 1MB.
- The SQL Selector operator allows one session; it does not support multiple sessions.
- The SQL Selector operator allows one instance; it does not support multiple instances.
- The SQL Selector operator is not restartable if the data export has started and the data export has not completed.
- The SQL Selector operator does not require a database load slot.

## Operational Considerations

### Checkpointing and Restarting

The SQL Selector operator takes a checkpoint only when all the data is sent to the Teradata PT data stream.

Then, on restart, the operator either sends none of the data, all of the data, or terminates with an error message, depending on the status of the data:

- If all of the data is sent, then the operator displays the following message and does not resend any of the data:

Restart indicates that this export job completed.

- If all of the data is not sent, then the operator terminates. Restart the job from the beginning.
- If none of the data is sent, then the operator sends the data.

# Stream Operator

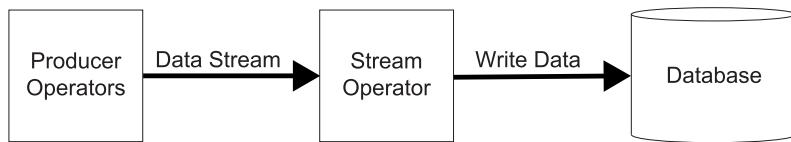
## Stream Operator Capabilities

The Stream operator, a consumer operator, emulates the Teradata TPump utility to perform high-speed parallel Inserts, Updates, Deletes, and Upserts in a near-real-time to one or more empty or preexisting database tables without locking target tables.

Row level locking, as used in the Stream operator's SQL transactions, allows load operations in the background during normal system use.

The Stream operator instances can update up to 127 tables on the database. Multiple parallel instances can be used to improve the performance of the update. For performance tuning in the Stream operator, see [Teradata PT Utility Commands](#).

The following figure shows the Stream operator interface.



## The Stream Operator Function in a Teradata PT Script

When you use the Stream operator in a Teradata PT job, the Stream operator:

1. Logs on to the database, using your user name, password, database name, and account ID information specified in the job script.
2. Loads the data from the Teradata PT data streams into the database.
3. Logs off from the database.
4. If the Stream operator job is successful, terminate the job and provide information about the job in the log or to the user, such as:
  - Total number of records sent to the database
  - Number of errors posted to the error table
  - Number of Inserts, Updates, and Deletes applied
5. If the job is unsuccessful, terminate the job and provide information about the job so that you can correct any problem and restart the job.

# TPump Utility and the Stream Operator

The following table lists the operating features and capabilities of the Teradata TPump utility and indicates whether they are supported by the Stream operator or another Teradata PT operator.

**Stream Operator Feature Support**

TPump Utility Feature	Stream Operator
Absolute field positioning (handled by the FIELD command)	Not supported
Access Modules	Supported, using the DataConnector operator
ANSI Date	Supported
ARRAY support for DML statements	Supported
Checkpoint/Restart	Supported
Character Sets	Supported, using the USING CHARACTER SET clause before the DEFINE JOB statement
Configuration File	Supported, using the <b>tbuild</b> command line job attribute option (-v)
CREATE TABLE Statement	Supported, using the DDL operator
DATABASE Statement	Supported
DELETE Statement	Supported, using the DDL operator
DROP TABLE Statement	Supported, using the DDL operator
Environment variable	Not supported
Error Limit	Supported
EXECUTE	Supported, using the APPLY Statement
IF-ELSE-ENDIF constructs	Not supported
Indicator Mode	Supported, using the DataConnector operator
INMOD Routines	Supported, via the MultiLoad INMOD Adapter operator
Maximum/Minimum Sessions	Supported
MARK/IGNORE EXTRA UPDATE/DELETE DML Option	Supported, using the MARK/IGNORE EXTRA UPDATE /DELETE DML option in the APPLY Statement
Nonindicator Mode	Supported, using the DataConnector operator
Notify	Supported
NULIF Clauses	Supported, but not the XB, XC, and XG data types
Operating System Command	Supported, using the OS Command operator

TPump Utility Feature	Stream Operator
Periodicity runtime parameter	Supported
Predefined system variables (i.e., SYSDATE, SYSDAY)	Not supported
QUEUETABLE	Supported
-m runtime parameter to keep macros	Supported
RATE keyword for limiting the rate at which statements are sent to the database	Supported
Record Formats	Supported, using the DataConnector operator
RECORD <i>n</i> THRU <i>m</i>	Supported in a limited form by the DataConnector operator, allowing you to read in the first <i>m</i> rows of a file, effectively allowing “RECORD 1 THRU <i>m</i> ”
Replication Services override	Supported
Routing of messages to alternate file	Supported via Logger Services
RUN FILE	Supported via Teradata PT script language
Schemas, more than one	Not supported
Show Version Information	Supported
Tenacity	Supported
TPump Monitor Interface	Not supported
User-defined variables	Limited support via script language
Wildcard INSERT	Not supported (cannot use “INSERT INTO tablename.*;”)

## Syntax

Use the following specifications and attributes to define the Stream operator in a Teradata PT job script.

## Required Specifications

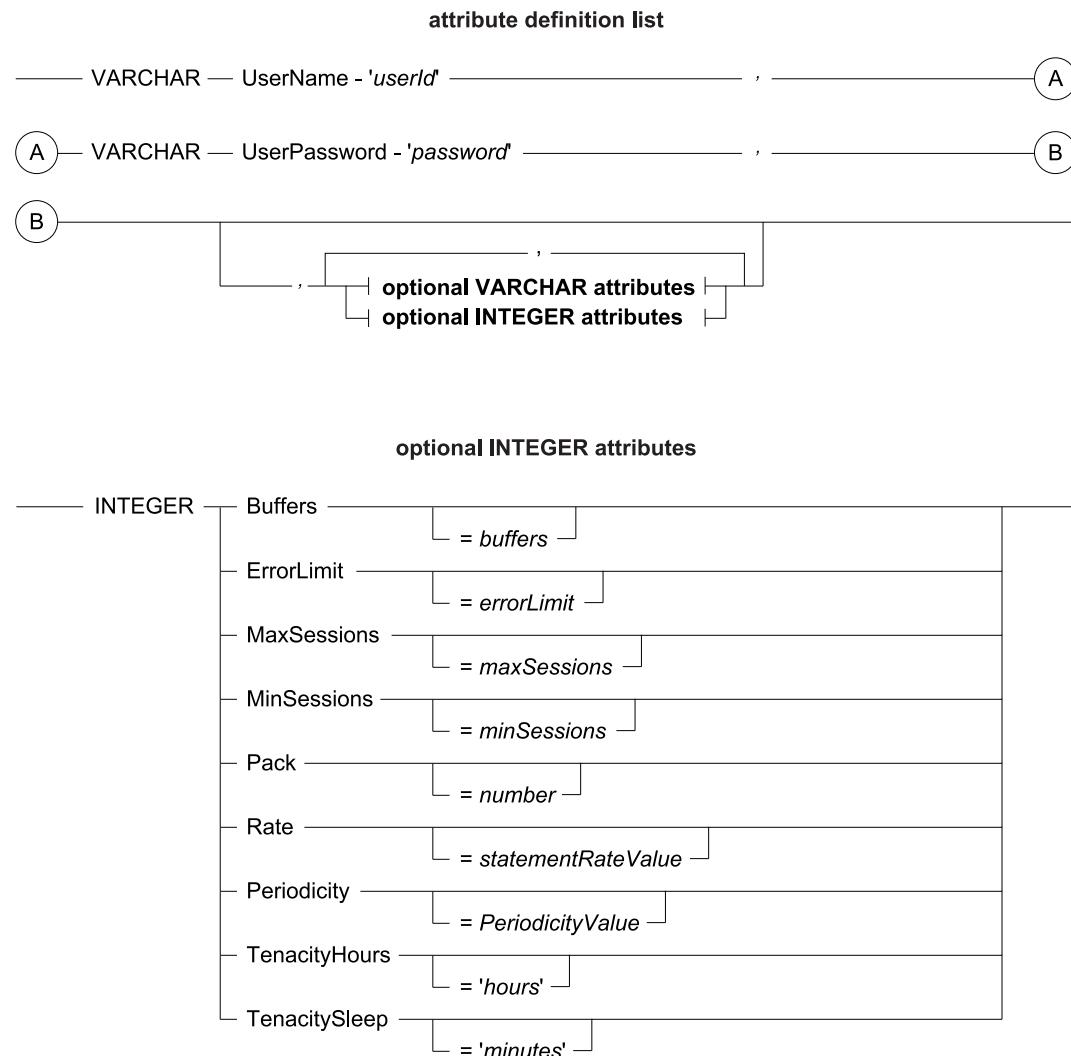
The following specifications are required to define a Stream operator job script.

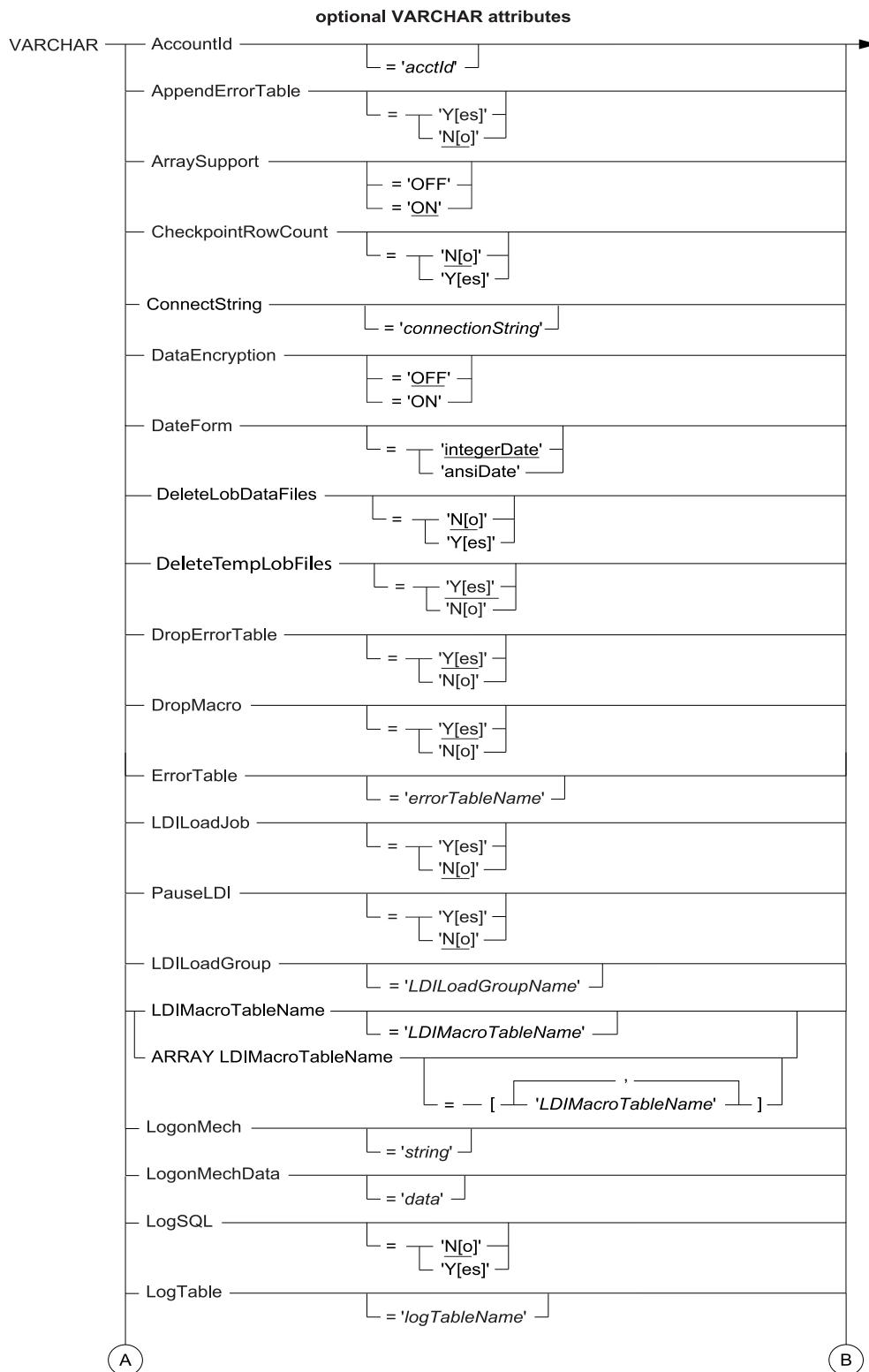
### Required Syntax for the Stream Operator

Specification	Description
TYPE	Operator type. Always STREAM for the Stream operator.
UserName	Operator attribute providing the name of the user for the Stream operator logon sessions.
UserPassword	Operator attribute providing the password associated with the user name.

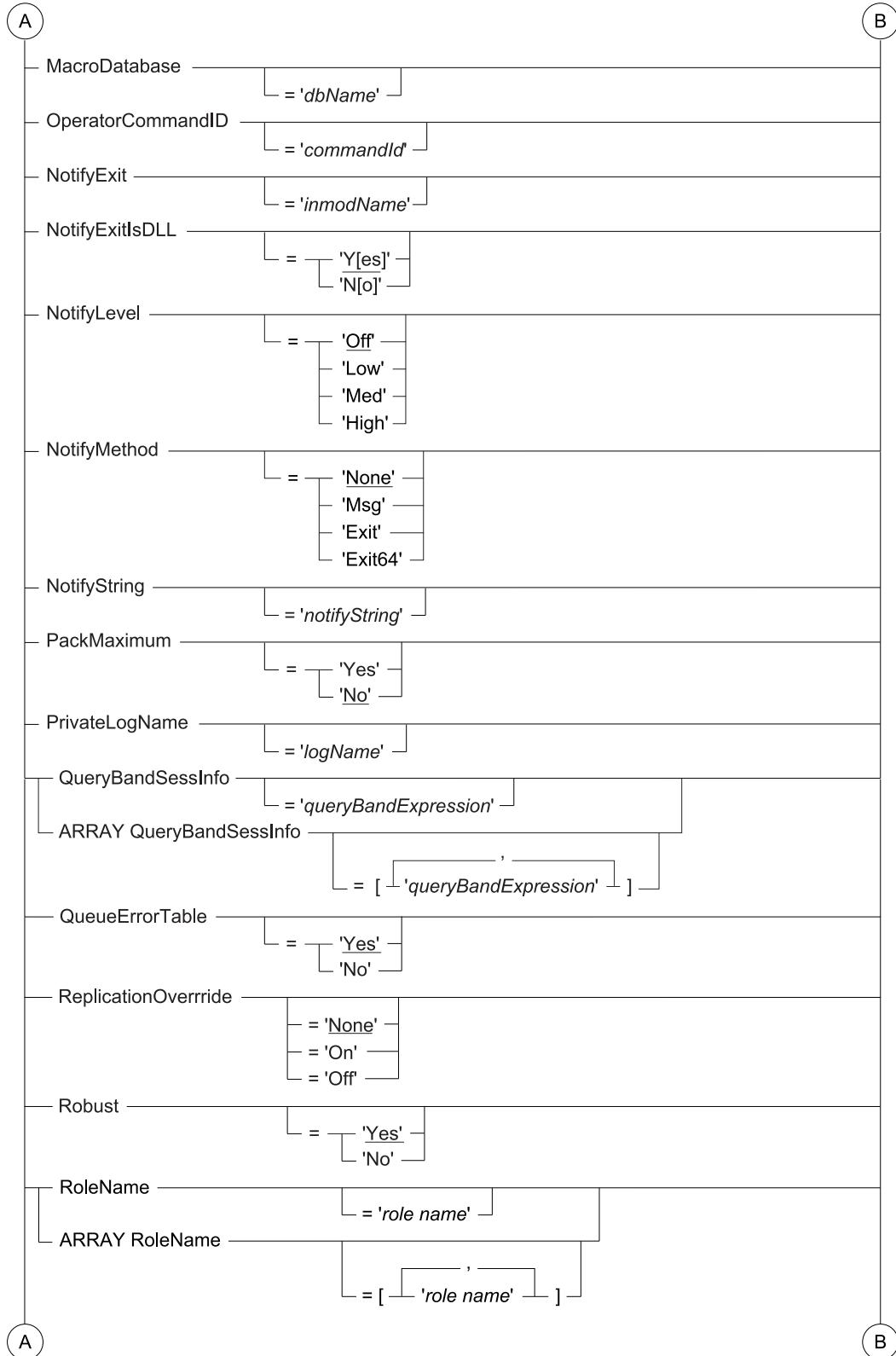
## Required and Optional Attributes

Use the attribute definition list syntax in the Teradata PT DEFINE OPERATOR statement to declare the required and optional attribute values for the Stream operator.

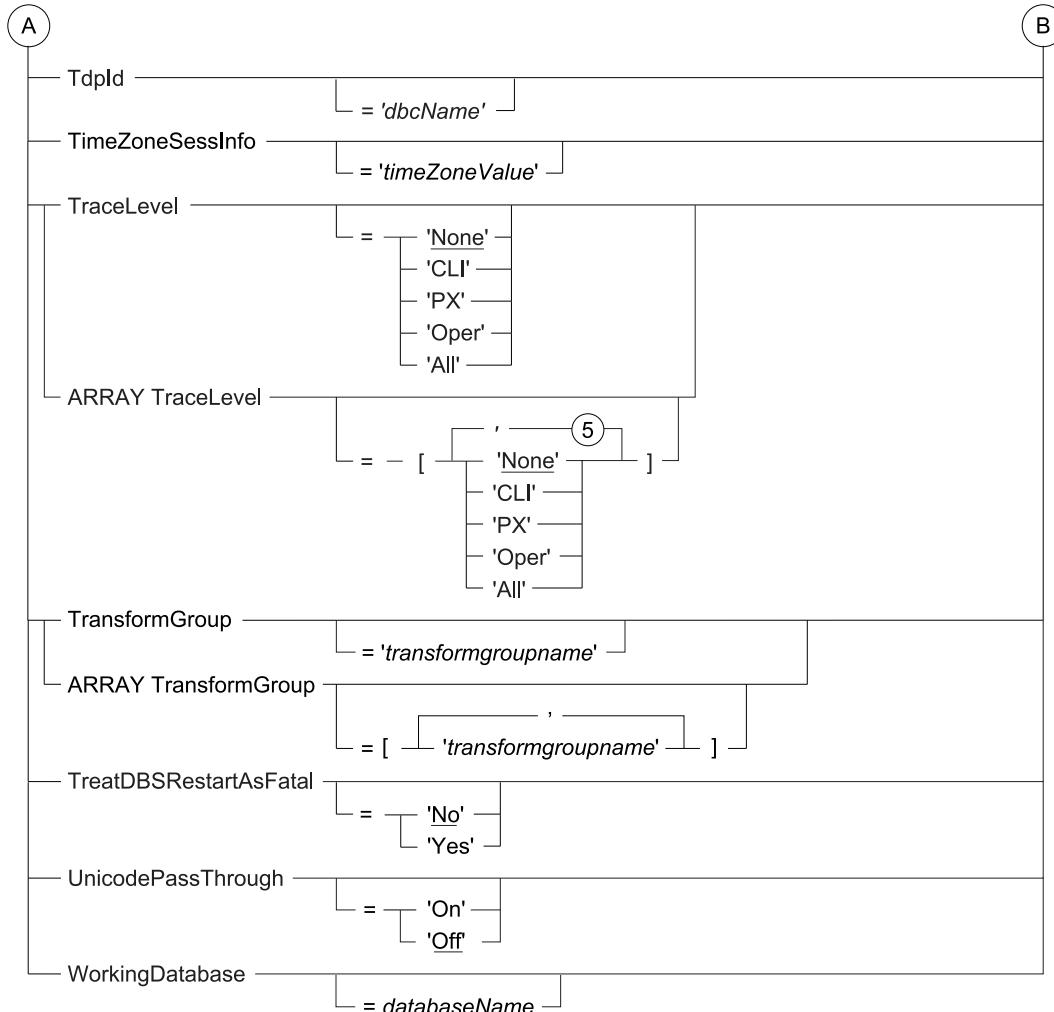




## optional VARCHAR attributes (continued)



## optional VARCHAR attributes (continued)



where:

Syntax Element	Description
AccountId = 'acctId'	Optional attribute that specifies the account associated with the user name. If omitted, it defaults to the account identifier of the immediate owner database.
AppendErrorTable = 'option'	Optional attribute that specifies whether the Stream Operator will use the existing error table. Valid values are: <ul style="list-style-type: none"> <li>'No' = Stream operator will not use the existing error table (default).</li> <li>'Yes' = Stream operator will use the existing error table or create the error table if it does not exist.</li> </ul> If the error table exists, the Stream operator displays the number of rows in the error table.

Syntax Element	Description
	<p>If the structure of the existing error table is not compatible with the error table that the Stream operator expects, the Stream operator terminates the job with an error message.</p> <p>By default, the Stream operator terminates the job with an error message if the error table already exists.</p> <p>On a job restart, the Stream operator will not terminate the job with an error message when the error table exists.</p>
ArraySupport = ' <i>option</i> '	<p>Optional attribute that specifies whether the Stream operator will use the ArraySupport feature for the job.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>• 'On' = Stream operator will use ArraySupport for the entire job.</li> <li>• 'Off' = Stream operator will not use ArraySupport.</li> </ul> <p>Even if the value for ArraySupport is not specified, Array Support is still enabled and the Stream operator will use it for the entire job if the following criteria are met:</p> <ul style="list-style-type: none"> <li>• Both the database and CLIV2 support the Array Support feature.</li> <li>• The DML statement is a single DML statement or an atomic UPSERT statement.</li> <li>• The job step must have a single DML group if Serialize is On.</li> </ul> <p>If any of these criteria are not met, then the default value is 'Off' and the Stream operator will not use Array Support.</p> <p>If the value for ArraySupport is set to 'On' and either the database or CLIV2 does not support the Array Support feature, the Stream operator will terminate with a fatal error.</p> <p>If the ARRAYSUPPORT DML option is used as part of the APPLY statement for a job, the DML value will override the value specified for the Stream operator ArraySupport attribute.</p>
Buffers = <i>buffers</i>	<p>Optional attribute that specifies whether to increase the number of request buffers.</p> <p>The range of values is a lower limit of 2 and no upper limit. The default value is 3. The maximum number of request buffers that may be allocated is the number of buffers multiplied by the number of connected sessions (Buffers * connected_sessions).</p> <p>Request buffers are a global resource, so buffers are assigned to any session as needed, and then returned to a free pool. At any point in time, the number of request buffers assigned to a session can vary from zero to Buffers * connected_sessions.</p>
CheckpointRowCount = ' <i>option</i> '	<p>Optional attribute that tells the Stream operator to enable or disable outputting the rows sent at checkpoints.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>• 'No' ('N') = The Stream operator will not output rows sent at checkpoints (default);</li> <li>• 'Yes' ('Y') = The Stream operator will output rows sent at checkpoints</li> </ul> <p>This attribute is only available in the TPT script mode.</p>
ConnectionString = ' <i>connectionString</i> '	Optional attribute that specifies the connection string. The connection string will be passed to CLI. CLI will validate the connection string.

Syntax Element	Description
	<p>For information on connection string, see <i>Teradata® Call-Level Interface Version 2 Reference for Workstation-Attached Systems</i>, B035-2418.</p> <p><b>Note:</b> The TPT Connection String feature is available on all platforms, except for the TDP variant of TPT on z/OS.</p>
DataEncryption =' <i>option</i> '	<p>Optional attribute that enables full security encryption of SQL requests, responses, and transmitted data.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>• 'On' = all SQL requests, responses, and data are encrypted.</li> <li>• 'Off' = no encryption occurs (default).</li> </ul>
DateForm = ' <i>option</i> '	<p>Optional attribute that specifies the DATE data type for the Stream operator job. The values are:</p> <ul style="list-style-type: none"> <li>• 'integerDate'= the integer DATE data type (default)</li> <li>• 'ansiDate'= the ANSI fixed-length CHAR(10) DATE data type</li> </ul>
DeleteLobDataFiles = ' <i>option</i> '	<p>Optional attribute that specifies whether to delete deferred LOB data files from the Data Connector Producer once the rows are committed to the database.</p> <p>Valid values for '<i>option</i>' are:</p> <ul style="list-style-type: none"> <li>• 'Yes' (or 'Y') = Delete deferred mode LOB data files once rows are committed to the database</li> <li>• 'No' (or 'N') = do not delete deferred mode LOB data files (default). Specifying any other value results in an error.</li> </ul>
DeleteTempLobFiles	<p>Optional attribute that tells the operator whether or not to delete the temporary LOB directory and the temporary LOB files in the directory at cleanup.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>• 'Y[es]' = Tells the operator to delete the temporary LOB directory at cleanup. This is the default.</li> <li>• 'N[o]' = Tells the operator not to delete the temporary LOB directory at cleanup. The user is responsible for deleting the temporary LOB directory.</li> </ul> <p><b>Note:</b> When the temporary LOB files exist in a remote NFS mount directory, the suggestion is to set this attribute to 'No' and the operator will not spend time deleting the temporary LOB files.</p>

Syntax Element	Description
	<p><b>Note:</b></p> <p>Temporary LOB files will be created in the temporary LOB directory. The naming convention for the temporary LOB directory will be as follows:</p> <ul style="list-style-type: none"> <li>When the producer's LobDirectoryPath attribute is set, the name of the temporary LOB directory will be this: &lt;LobDirectoryPath&gt;/&lt;job_id&gt;_s&lt;job step number&gt;_TempLOBDir</li> <li>When the producer's LobDirectoryPath attribute is not set, the name of the temporary LOB directory will be this: &lt;current working dir&gt;/&lt;job_id&gt;_s&lt;job step number&gt;_TempLOBDir</li> </ul> <p>This attribute is not supported in TPTAPI.</p> <p>This attribute is not supported on z/OS.</p>
DropErrorTable = ' <i>option</i> '	<p>Optional attribute that specifies whether the Stream Operator will drop the error table at the end of a job.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>'Yes' = Stream operator drops the error table when it is empty at the end of a job (default). Teradata PT automatically executes a DROP TABLE statement.</li> <li>'No' = Stream operator will not drop the error table even if it is empty at the end of a job.</li> </ul> <p>If you run many small (short-duration) load jobs on a regular basis, setting DropErrorTable to 'No' for these jobs reduces updates to the Vantage data dictionary.</p> <p>If the error table is not dropped, it can be used when AppendErrorTable is set to 'Yes' at the beginning of the next job.</p>
DropMacro = ' <i>option</i> '	<p>Optional attribute that instructs the Stream operator whether to drop macros or keep them for future use.</p> <pre>VARCHAR DropMacro = '&lt;Y[es] N[o]&gt;'</pre> <p>By default, the Stream operator drops macros at the end of a successful job. When the value of DropMacro is N or No, the macros remain in the database until a DROP MACRO statement is issued against them. Kept macros are maintained by the database until they are explicitly dropped, so be sure to check for obsolete macros periodically and drop them.</p>
ErrorLimit = <i>limit</i>	<p>Optional attribute that the database specifies the approximate number of records that can be stored in the error table before the Stream operator job is terminated.</p> <p>This number is approximate because the Stream operator sends multiple rows of data at a time to the database. By the time Teradata PT processes the message indicating that the error limit has been exceeded, it may have loaded more records into the error table than the actual number specified in the error limit.</p> <p>The ErrorLimit specification must be greater than 0. Specifying an invalid value will cause the Stream operator job to terminate. By default, the ErrorLimit value is unlimited.</p> <p><b>Note:</b></p> <p>The ErrorLimit specification applies to each instance of the Stream operator.</p>

Syntax Element	Description
ErrorTable = 'errorTableName'	<p>Optional attribute that specifies the name of the error table. This table contains information concerning data conversion errors, constraint violations, and other error conditions.</p> <p>Prefix the error table name with a database name as a qualifier. This means that because the database might contain a lot of PERM space, that space does not need to be increased for all databases with tables involved in the load.</p> <p>If the database for the error table is not specified, the table is placed in the database associated with the user logon.</p> <p>By default, the error table must be a new table. This default can be changed. If the AppendErrorTable attribute is set to 'Yes' or 'Y', then the error table can be a new or existing table.</p> <p>Do not use a name that duplicates the name of an existing table unless you are restarting a Stream Operator job or reusing the error table by setting the AppendErrorTable attribute to 'Yes' or 'Y'.</p> <p>If the name is not supplied, it is created by the Stream Operator.</p> <p>User-supplied names for error tables must not exceed the maximum allowable size for table names on the database.</p>
LDILoadGroup = 'value'	<p>Optional attribute that tells the Stream operator to LDILoadGroup value. This value is required if LDILoadJob is set to 'Yes' ('Y').</p> <p>For example, LDILoadGroup = 'Grp1'</p>
LDILoadJob = 'option'	<p>Optional attribute that tells the Stream operator to enable or disable the Isolated Load feature.</p> <p>The LDILoadJob values are:</p> <ul style="list-style-type: none"> <li>'No' ('N') = No Isolated Load (default)</li> <li>'Yes' ('Y') = Request Isolated Load. The stream operator will send "BEGIN ISOLATED LOADING ON &lt;target tables&gt; USING QUERY_BAND 'LDILoadGroup=&lt;...&gt;'; IN MULTIPLE SESSION;" request to the database if the target tables are qualified. The target tables have to be created as "WITH ISOLATED LOADING FOR ALL".</li> </ul> <p>For example:</p> <pre>CREATE TABLE testtable,     FALBACK,     WITH ISOLATED LOADING FOR ALL     (COL1      VARCHAR(5),      COL2      VARCHAR(5));';</pre>
LDIMacroTableName = ['table1', 'table2' ...]	<p>Optional attribute that specifies the LDI table names for the predefined macros. It is required when predefined macros are used, LDILoadJob is set to 'Y[es]' and LDILoadGroup is specified with a value.</p>
LogonMech = 'string'	<p>Optional attribute that specifies which logon mechanism to use.</p> <p><b>Note:</b></p> <p>Specification of this attribute may be required for some authentication methods. The job terminates if the attribute exceeds 8 bytes.</p> <p>For information on specification requirements for LogonMech, see "Logon Security" in <i>Teradata® Parallel Transporter User Guide</i>, B035-2445.</p>

Syntax Element	Description
LogonMechData = ' <i>data</i> '	<p>Optional attribute that passes along additional logon data.</p> <p><b>Note:</b> Specification of this attribute is required for some external authentication methods. For information on specification requirements for LogonMechData, see “Logon Security” in <i>Teradata® Parallel Transporter User Guide</i>, B035-2445.</p>
LogSQL = ' <i>option</i> '	<p>Optional attribute that controls how much of the job’s SQL to enter into the log.</p> <p>Valid options are:</p> <ul style="list-style-type: none"> <li>• 'Yes' = output the full SQL to the log. The maximum length is 1M.</li> <li>• 'No' = do not output SQL to the log.</li> <li>• No value or attribute omitted = accept the predefined limit, which displays up to 32K of SQL if all the SQL is less than 32K. If the SQL to be logged exceeds 32K, truncate the display to the first 32K bytes.</li> </ul>
LogTable = ' <i>logTableName</i> '	<p>Optional attribute that specifies the name of the restart log table for checkpoint information. The restart log table must be a unique table name and must not exist, unless restarting from a paused job.</p> <p>If the restart log table does not exist, the Stream operator creates it. If it exists, the Stream operator restarts from the last checkpoint. Failure to specify a restart log table terminates the job.</p> <p>The following privileges are required on the restart log table:</p> <ul style="list-style-type: none"> <li>• SELECT</li> <li>• INSERT</li> <li>• DELETE</li> </ul> <p>The following privileges are required on the database that contains the restart log table:</p> <ul style="list-style-type: none"> <li>• DROP</li> <li>• CREATE</li> </ul> <p>The Stream operator automatically maintains the restart log table. Manipulating the restart log table in any way invalidates the restart capability. If the restart log table name is not fully qualified, it is created under the user’s default (logon) database. If the WorkingDatabase attribute is used, you MUST fully qualify the restart log table name, even if the restart log table is going to reside in the default (logon) database.</p>
MacroDatabase = ' <i>dbName</i> '	<p>Optional attribute that specifies the database to contain any macros used by the Stream operator.</p> <p>The default macro database is the restart log table database.</p>
MaxSessions = <i>maxSessions</i>	<p>Optional attribute that specifies the maximum number of sessions to log on. The MaxSessions value must be greater than 0. Specifying a value less than 1 terminates the job.</p> <p>The default is one session for each operator instance.</p> <p>The main instance calculates an even distribution of the Stream operator sessions among the number of instances. For example, if there are 4 instances and 16 Stream operator sessions, then each instance will log on 4 Stream operator sessions.</p>

Syntax Element	Description
MinSessions = <i>minSessions</i>	<p>Optional attribute that specifies the minimum number of sessions required for the Stream operator job to continue.</p> <p>The MinSessions value must be greater than 0 and less than or equal to the maximum number of Stream operator sessions. Specifying a value less than 1 terminates the job.</p> <p>The default is 1.</p>
NotifyExit = ' <i>inmodName</i> '	<p>Attribute that specifies the name of the user-defined notify exit routine with an entry point named _dynamn. If no value is supplied, the following default name is used:</p> <ul style="list-style-type: none"> <li>• libnotfyext.dll for Windows platforms</li> <li>• libnotfyext.dylib for the Apple macOS platform</li> <li>• libnotfyext.so for all other UNIX platforms</li> <li>• NOTFYEXT for z/OS platforms</li> </ul> <p>See <a href="#">Deprecated Syntax</a> for information about providing your own notify exit routine.</p>
NotifyLevel = ' <i>notifyLevel</i> '	<p>Optional attribute that specifies the level at which certain events are reported. The valid values are:</p> <ul style="list-style-type: none"> <li>• 'Off' = no notification of events is provided (default)</li> <li>• 'Low' = 'Yes' in the Low Notification Level column</li> <li>• 'Med' = 'Yes' in the Medium Notification Level column</li> <li>• 'High' = 'Yes' in the High Notification Level column</li> <li>• "Ultra" = 'Yes" in the Ultra Notification Level column</li> </ul>
NotifyMethod = ' <i>notifyMethod</i> '	<p>Optional attribute that specifies the method for reporting events. The methods are:</p> <ul style="list-style-type: none"> <li>• 'None'= no event logging is done (default).</li> <li>• 'Msg'= sends the events to a log.</li> </ul> <p>On Windows, the events are sent to the event log that can be viewed using the Event Viewer. The messages are sent to the Application log.</p> <p>On Solaris, AIX, and Linux platforms, the destination of the events is dependent upon the setting specified in the /etc/syslog.conf file.</p> <p>On SLES11, the destination of the events is dependent upon the setting specified in the /etc/syslog-ng.conf file.</p> <p>On z/OS systems, events are sent to the job log.</p> <ul style="list-style-type: none"> <li>• 'Exit'= sends the events to a user-defined notify exit routine. Row count information is in 4-byte unsigned integer values.</li> <li>• 'Exit64' = sends the events to a user-defined notify exit routine. Row count information is in 8-byte unsigned integer values for these events: NMEventCkptBeg64 NMEventCkptEnd64 NMEventErrorTable64 NMEventRunStats64 NMEventImportEnd64</li> <li>• 'ExitEON' = sends the events to a user-defined notify exit routine. Complete Teradata object names are passed to the notify exit routine for the NMEventInitializeEON event.</li> </ul>

Syntax Element	Description
	In addition ExitEON sends row count information in 8-byte unsigned integer values.
NotifyString = ' <i>notifyString</i> '	<p>Optional attribute that provides a user-defined string to precede all messages sent to the system log. This string is also sent to the user-defined notify exit routine. The maximum length of the string is:</p> <ul style="list-style-type: none"> <li>• 80 bytes, if the NotifyMethod is 'Exit'</li> <li>• 16 bytes, if NotifyMethod is 'Msg'</li> </ul>
OperatorCommandID = ' <i>commandId</i> '	<p>Although you can specify rate and periodicity values using Stream operator attributes, you may not know the optimal values for a specific job step until after the job has begun running. OperatorCommandId allows you to identify a specific reference of a Stream operator to which you can assign new rate or periodicity values after the job has begun, using twbcmd:</p> <p>Teradata PT will generate a default value for OperatorCommandId composed of <i>&lt;operator object name&gt; + &lt;process Id&gt;</i> for each copy of the operator in the APPLY specification. If you want to assign another identifier, do the following:</p> <p>Declare the operatorCommandId attribute in the DEFINE OPERATOR statement for the Stream operator.</p> <p>You can optionally assign a value to the OperatorCommandID attribute in a referenced copy of the Stream operator (in an APPLY statement). If no value is assigned, Teradata PT will provide a system-generated value. A useful OperatorCommandID value might be the number of the job step in which you want to change the Rate, as follows:</p> <pre>APPLY &lt;dm11&gt; TO OPERATOR ( Stream_Oper[2] ATTRIBUTES ( OperatorCommandID = 'ratestep#1' ), APPLY &lt;dm12&gt; TO OPERATOR ( Stream_Oper[3] ATTRIBUTES ( OperatorCommandID = 'ratestep#2' ) ),</pre> <p>Use the twbcmd utility to assign a Rate value to a specific Stream operator copy.</p> <p>For information on use of the twbcmd utility to change the Rate value, see <a href="#">twbcmd</a>.</p>
Pack = <i>number</i>	<p>Optional attribute that specifies the number of statements to pack into a multiple statement request. The maximum value is 2400. The default value is 20.</p> <p><b>Note:</b></p> <p>When the job is loading 1 or more deferred LOB/JSON/XML columns, the largest possible pack factor is 4096 because the database can support up to 4096 spool files per request.</p>
PackMaximum = ' <i>option</i> '	<p>Optional attribute that requests the Stream operator to dynamically determine the maximum possible pack factor for the current Stream job. The PackMaximum values are:</p> <ul style="list-style-type: none"> <li>• 'No' ('N') = no pack (default)</li> <li>• 'Yes' ('Y') = determine maximum possible pack factor</li> </ul>

Syntax Element	Description
PauseLDI =' <i>option</i> '	<p>Optional attribute that tells the Stream operator to pause the Isolated Load feature.</p> <p>The PauseLDI values are:</p> <ul style="list-style-type: none"> <li>'No' ('N') = The Stream operator will send the "END ISOLATED LOADING FOR QUERY_BAND 'LDILoadGroup=&lt;value&gt;;'" to the database if "BEGIN ISOLATED LOADING ON &lt;target table(s)&gt; USING QUERY_BAND 'LDILoadGroup=&lt;value&gt;;'" has been sent.</li> <li>'Yes' ('Y') = The Stream operator will not send the "END ISOLATED LOADING FOR QUERY_BAND 'LDILoadGroup=&lt;value&gt;;'" to the database.</li> </ul>
Periodicity = <i>periodicity</i>	<p>Option that specifies that the DML statements sent by the Stream operator to the database will be as evenly distributed as possible over each one minute interval. The periodicity value sets the number of sub-intervals per minute. Periodicity facilitates the orderly and efficient use of system resources.</p> <p>For example: If the statement rate is 1600 and the periodicity value is 10, then the maximum number of statements processed is 160 (1600/10) statements every 6 (60/10) seconds.</p> <p>Use of the Periodicity attribute is subject to the following conditions and rules:</p> <ul style="list-style-type: none"> <li>The valid values are integers between 1 and 600.</li> <li>The default value is 4, which means four 15-second periods per minute.</li> <li>If the statement rate is unlimited, then the Periodicity value is ignored.</li> <li>While the job is running, users can change the Periodicity value using the Teradata PT External command interface utility twbcmd.</li> </ul>
PrivateLogName = ' <i>logName</i> '	<p>Optional attribute that specifies the name of a log that is maintained by the Teradata PT Logger inside the public log. The private log contains all the output provided by the Stream operator.</p> <p>The private log can be viewed using the tlogview command as follows, where <i>jobid</i> is the Teradata PT job name and <i>privateLogName</i> is the value for the Stream operator PrivateLogName attribute:</p> <pre>tlogview -j jobid -f privateclassname</pre> <p>If the private log is not specified, all the output is stored in the public log. By default, no diagnostic trace messages are produced. Diagnostic trace messages are produced only when the user sets a valid value for the TraceLevel attribute.</p> <p>For more information about the tlogview command, see <a href="#">Teradata PT Utility Commands</a>.</p>
QueryBandSessInfo = ' <i>queryBandExpression</i> '	<p>Optional attribute that specifies the Query Band for the duration of the job sessions.</p> <p>The <i>queryBandExpression</i> is a set of name=value pairs, separated by a semicolon and ending with a semicolon. The user defines the Query Band expression, which will look similar to the following example:</p> <pre>'org=Finance;load=daily;location=west;'</pre> <p>QueryBandSessInfo may also be specified as an ARRAY attribute.</p>

Syntax Element	Description
	<p>For information on the rules for creating a Query Band expression, see <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i>, B035-1144 and <i>Teradata Vantage™ - SQL Data Definition Language Detailed Topics</i>, B035-1184.</p> <p>The value of the <code>QueryBandSessInfo</code> attribute is displayed in the Stream operator private log.</p> <p>Use of the <code>QueryBandSessInfo</code> attribute is subject to the following rules:</p> <ul style="list-style-type: none"> <li>• By default, Query Band is off until a valid value appears for the <code>QueryBandSessInfo</code> attribute.</li> <li>• If the <code>QueryBandSessInfo</code> attribute contains a value, the Stream operator constructs the necessary SET QUERY BAND SQL and issues it as part of the Stream operator SQL sessions to communicate the request to the database.</li> <li>• The Stream operator does not check the Query Band expression, but passes the expression to the database as is.</li> <li>• If the version of the database against which the job is being run does not support the Query Band feature, no Query Banding will take place. However, the operator will ignore the error and run the rest of the job.</li> <li>• If there is a syntax error in the Query Band expression, the database will return an error. The Stream operator will then terminate the job and report the error to the user.</li> </ul>
<code>QueueErrorTable = 'option'</code>	<p>Optional attribute that specifies whether the error table is a queue table.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>• 'Yes' ('Y') = create the error table as a queue table.</li> <li>• 'No' ('N') = create the error table as a non-queue table (default).</li> </ul> <p>This attribute is unique to the Stream operator and to the error table in the Stream operator. This attribute is especially useful in capturing errors that result when using the SELECT and CONSUME database operation, which returns rows and DELETEs them. Using an error table as a queue table can eliminate the need to delete the rows in the error table.</p>
<code>Rate = statementRate</code>	<p>Option that specifies the maximum number of DML statements per minute the Stream operator can submit to the database.</p> <p>Use of the Rate attribute is subject to the following conditions and rules:</p> <ul style="list-style-type: none"> <li>• The statement rate must be a positive integer.</li> <li>• If the statement rate is not specified, the rate is unlimited.</li> <li>• If the statement rate is less than the statement packing factor, the Stream operator sends requests smaller than the packing factor.</li> <li>• If the statement rate is invalid, Stream Operator will display an error message and terminate with a return code of 8.</li> <li>• If Serialization is not in use: the Stream operator will add successive rows to the session input “bucket” until the Pack value is reached. Then it will send the data and go on to the next session in the session list and begin filling the new input bucket. However, if statement rate/periodicity (R/P) is less than the packing factor, then only R/P rows will be added to a session bucket before the contents of the bucket are sent to the database.</li> <li>• If Serialization is in use: The data in the serialization columns is hashed and the resulting value is used as an index to the session list to select the session for the data row. Thus when using Serialization and R/P is less</li> </ul>

Syntax Element	Description
	<p>than the Pack factor, the R/P rows sent to the database may be scattered across multiple sessions.</p> <ul style="list-style-type: none"> <li>While the job is running, users can change the statement rate value using the Teradata PT External Command interface utility <code>twbcmd</code>.</li> </ul>
ReplicationOverride = ' <i>option</i> '	<p>Optional attribute that overrides the normal replication services controls for an active session.</p> <p>Valid values:</p> <ul style="list-style-type: none"> <li>'On' = override normal replication services controls for the active session.</li> <li>'Off' = override of normal replication services is turned off for the active session (when change data capture is active).</li> <li>'None' = no override request is sent to the database (default)</li> </ul> <p>For more information, see <i>Teradata Replication Services Using Oracle GoldenGate</i>.</p> <p><b>Note:</b></p> <p>The user ID that is logged in by the operator must have the REPLCONTROL privilege when setting the value for this attribute.</p>
Robust = ' <i>option</i> '	<p>Optional attribute that specifies whether to use robust restart logic for recovery/restart operations.</p> <p>In "robust mode," one database row is written in the log restart table for every request issued. This collection of rows in the restart log table can be referred to as the request log. Because a request is guaranteed by the database to either completely finish or completely roll back, the request log will always accurately reflect the completion status of an import.</p> <p>The Robust values are:</p> <ul style="list-style-type: none"> <li>'Yes' ('Y') = Use robust restart logic (default). In the robust mode, for each packed request, several "partial checkpoint" rows are written to the log between checkpoints. The rows are deleted each time a checkpoint is written. In Robust recovery mode, the Stream operator must next ascertain how much processing has been completed since the last logged checkpoint. This is accomplished by reading back a set of "Partial Checkpoints" from the database, sorting them and then reprocessing all transactions that were left incomplete when the job was interrupted.</li> <li>'No' ('N') = Use simple restart logic. In this case, restarts cause the Stream operator to begin where the last checkpoint occurs in the job. Any processing that occurs after the checkpoint is redone. This method does not have the extra overhead of the additional database writes in the robust logic, and should be adequate in certain DML statements that can be repeated without changing the results of the operation.</li> </ul> <p>If uncertain whether to use robust restart logic, it is always safe to set the Robust parameter to 'Yes'.</p>
RoleName = ' <i>role name</i> '	<p>Optional attribute that implements security in the database. The operator will prepend the value with "SET ROLE ". The syntax will be sent to the database as follows:</p> <pre>SET ROLE &lt;i&gt;role name&lt;/i&gt;;</pre> <p>For example:</p> <pre>SET ROLE All;</pre>

Syntax Element	Description
	<p>For details of "SET ROLE" command use, see <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i>, B035-1144.</p> <p>The VARCHAR ARRAY can specify more than one value, for example:</p> <pre>VARCHAR ARRAY RoleName = [ 'role name1', 'role name2' ],</pre> <p>The operator will send the request to the database on the main control and data SQL sessions after the sessions are connected.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>• The operator does not validate the value for this attribute. The operator passes the value to the database as is. The database will validate the value. The operator will terminate the job with an error when the validation fails.</li> <li>• C-style comments are allowed in the value and will be passed to the database.</li> <li>• ANSI-style comments are not supported in the value. The operator can terminate the job with a syntax error when the value contains an ANSI-style comment.</li> <li>• A semicolon is not allowed in the value, because the operator allows only a single statement per request. The operator will terminate the job with an operator error when the value contains a semicolon.</li> </ul>
TdplId = ' <i>dbcName</i> '	<p>Optional attribute that specifies the name of the database machine (non-mainframe platforms) or TDP (mainframe platforms) for the Stream operator job.</p> <p>The <i>dbcName</i> can be up to 256 characters and can be a domain server name.</p> <p>If you do not specify the value for the TdplId attribute, the operator uses the default TdplID established for the user by the system administrator.</p>
TenacityHours = <i>hours</i>	<p>Optional attribute that specifies the number of hours that the Stream operator continues trying to log on when the maximum number of sessions are already running on the database.</p> <p>The default value is 4 hours. To enable the tenacity feature, the <i>hours</i> value must be greater than 0. Specifying a value of 0 disables the tenacity feature. Specifying a value of less than 0 terminates the Stream operator job.</p>
TenacitySleep = <i>minutes</i>	<p>Optional attribute that specifies the number of minutes that the Stream operator job pauses before retrying a log on operation when the maximum number of sessions are already running on the database.</p> <p>The <i>minutes</i> value must be greater than 0. If you specify a value less than 1, the Stream operator responds with an error message and terminates the job. The default is 6 minutes.</p>
TimeZoneSessInfo = ' <i>timeZoneValue</i> '	<p>Optional attribute that allows you to change the default time zone displacement for the duration of the operator's job session.</p> <p>When you provide a value for this attribute, the operator will build the SET TIME ZONE &lt;<i>timeZoneValue</i>&gt;; SQL request.</p> <p>The operator will send the request to the database on the main control and data SQL sessions after the sessions are connected.</p> <p>Here are some examples:</p>

Syntax Element	Description
	<ul style="list-style-type: none"> <li><b>Example 1:</b> This example sets the session default time zone displacement to LOCAL, which is the system default time zone:  <code>VARCHAR TimeZoneSessInfo = 'LOCAL'</code></li> <li><b>Example 2:</b> This example sets the session default time zone displacement to USER, which is the default time zone for the logged on user:  <code>VARCHAR TimeZoneSessInfo = 'USER'</code></li> <li><b>Example 3:</b> This example sets the session default time zone displacement to a simple constant time zone string expression:  <code>VARCHAR TimeZoneSessInfo = '''America Pacific'''</code></li> </ul> <p><b>Note:</b>  Any single quote character ('') inside the value must be entered as two consecutive single quote characters in a TPT job script. This ensures the correct value will be sent to the database.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>The operator does not validate the value for this attribute. The operator passes the value to the database as is. The database will validate the value. The operator will terminate the job with an error when the validation fails.</li> <li>C-style comments are allowed in the value and will be passed to the database.</li> <li>ANSI-style comments are not supported in the value. The operator will terminate the job with a syntax error when the value contains an ANSI-style comment.</li> <li>A semicolon is not allowed in the value, because the operator allows only a single statement in the "SET TIME ZONE SQL" request. The operator will terminate the job with an operator error when the value contains a semicolon.</li> </ul> <p>For more information on SET TIME ZONE SQL, see <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i>, B035-1144.</p>
TraceLevel = 'level'	<p>Optional attribute that specifies the types of diagnostic messages that are written by each instance of the operator to the public log (or private log, if one is specified using the PrivateLogName attribute). The diagnostic trace function provides more detailed information in the log file to aid in problem tracking and diagnosis. The trace levels are:</p> <ul style="list-style-type: none"> <li>'None'= TraceLevel turned off (default).</li> <li>'CLI'= enables the tracing function for CLI-related activities (interaction with the database)</li> <li>'PX'= enables the tracing function for activities related to the Teradata PT infrastructure</li> <li>'Oper'= enables the tracing function for operator-specific activities</li> <li>'Notify'= enables the tracing function for activities related to the Notify feature</li> <li>'All'= enables tracing for all the previous activities</li> </ul> <p>The VARCHAR ARRAY can specify more than one value, for example:</p>

Syntax Element	Description
	<pre>VARCHAR TraceLevel = 'CLI' VARCHAR TraceLevel = 'OPER' VARCHAR ARRAY TraceLevel = [ 'CLI' ] VARCHAR ARRAY TraceLevel = [ 'CLI', 'OPER' ]</pre> <p><b>Note:</b> The TraceLevel attribute is provided as a diagnostic aid only. The amount and type of additional information provided by this attribute will change to meet evolving needs from release to release.</p>
TransformGroup = 'transformgroupname'	<p>Optional attribute that supports changing the active transform for Teradata Complex Data Types (CDTs). The value is the <i>&lt;udt name&gt; &lt;transform group name&gt;</i>, and the operator will prepend the hardcoded string "SET TRANSFORM GROUP FOR TYPE ". The syntax sent to RDBMS is as follows:</p> <pre>SET TRANSFORM GROUP FOR TYPE &lt;udt name&gt; &lt;transform group name&gt;;</pre> <p>For example:</p> <pre>"SET TRANSFORM GROUP FOR TYPE JSON CHARACTER SET LATIN TD_JSON_VARCHAR;"</pre> <p>The VARCHAR ARRAY can specify more than one value, for example:</p> <pre>VARCHAR ARRAY TransformGroup = [ 'JSON CHARACTER SET LATIN TD_JSON_VARCHAR',                                 'ST_GEOGRAPHY TD_GEO_VARCHAR' ],</pre> <p>The operator will send the request to the database on the main control and data SQL sessions after the sessions are connected.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>The operator does not validate the value for this attribute. The operator passes the value to the database as is. The database will validate the value. The operator will terminate the job with an error when the validation fails.</li> <li>C-style comments are allowed in the value and will be passed to the database.</li> <li>ANSI-style comments are not supported in the value. The operator can terminate the job with a syntax error when the value contains an ANSI-style comment.</li> <li>A semicolon is not allowed in the value, because the operator allows only a single statement per request. The operator will terminate the job with an operator error when the value contains a semicolon.</li> </ul>
TreatDBSRestartAsFatal= 'option'	<p>Optional attribute that tells the operator whether to terminate the job when a database restart occurs.</p> <p>The TreatDBSRestartAsFatal values are:</p> <ul style="list-style-type: none"> <li>'No' ('N') = The operator will not terminate if a database restart occurs (default). The database restart will be treated as a retryable one.</li> <li>'Yes' ('Y') = The operator will terminate if a database restart occurs.</li> </ul>

Syntax Element	Description
UnicodePassThrough = 'value'	<p>Optional attribute that tells the operator to enable or disable the Unicode Pass Through feature.</p> <p>Valid values:</p> <ul style="list-style-type: none"> <li>• 'On' = Enable the Unicode Pass Through feature in the operator.</li> <li>• 'Off' = (Default) Disable the Unicode Pass Through feature in the operator.</li> </ul> <p><b>Note:</b></p> <p>When a TPT job is using the UTF8 or UTF16 session character set, the UnicodePassThrough attribute can be set to 'On' to allow the operator to load data with Unicode pass through characters.</p>
UserName = 'userId'	<p>Attribute that specifies the user name.</p> <p><b>Note:</b></p> <p>Use of this attribute is not compatible with some external authentication logon methods.</p>
UserPassword = 'password'	<p>Attribute that specifies the password associated with the user name.</p> <p><b>Note:</b></p> <p>Use of this attribute is not compatible with some external authentication logon methods.</p>
WorkingDatabase = 'databaseName'	<p>Optional attribute that specifies a database other than the logon database as the default database.</p> <p>The name of the database that is specified with this attribute is used in the Teradata SQL DATABASE statement that is sent by the operator immediately after connecting the two SQL sessions.</p> <p>If WorkingDatabase is not specified, the default database associated with the logged on user is assumed for all unqualified table names.</p>

## Usage Notes

### LogTable

A restart log table, which contains restart information written during job runs, is required for any execution of the Stream operator. Specify a restart log table in scripts with the LogTable attribute.

Restarts can occur following any unexpected error on a database. For example, if a table runs out of available space during a load operation, the job terminates, the table is paused, and a checkpoint is recorded in the restart log. Pausing the job in this way allows you to manually increase the available space, if needed, then restart the job because the load operation can restart a job from the last checkpoint in the restart log.

If you do not specify a name for *LogTable*, the Stream operator automatically creates a name of the log table as follow:

```
ttname_RL
```

where *ttname* is the name of the corresponding target table.

**Note:**

The value of the TargetTable attribute is truncated to the maximum number of characters for object names that the database supports, minus 3 characters before the suffix "\_RL" is appended to the target table name. This means that if the value of the TargetTable attribute is a fully qualified table name and that fully qualified name exceeds the maximum supported length of a database object, the generated name for the log table may not be what you intend. In such a case, Teradata recommends that you provide the names of the log table and not rely on the Stream operator to generate the names for log table automatically.

## ErrorTable

Attribute that specifies the name of the error table. This table contains information concerning data conversion errors, constraint violations, and other error conditions.

This attribute must be a new table name, unless the AppendErrorTable attribute is set to YES. Do not use a name that duplicates the name of an existing table unless you are restarting a Stream operator job. If the name is not supplied, it is created by the Stream operator. User-supplied names for error tables must not exceed the maximum size for table names on the database.

## Keeping the Error Table

By default, the Stream operator drops the error table after data loading, unless the error table is non-empty. The error table can be kept after data loading by setting the DropErrorTable attribute to NO, as follows:

```
VARCHAR DropErrorTable = 'NO'
```

Keeping the error table allows the error table to be re-used for future jobs and saves the time required to drop the table.

## Re-using the Error Table

By default, the Stream Operator will create the error table. If the error table already exists, Stream Operator will terminate the job with an error message. The exception is on a job restart. On a job restart, Stream Operator will continue.

- When the value of the AppendErrorTable attributes is YES, the Stream operator reuses the error table and displays the number of rows already existing in the error table.

- If the error table does not exist, the Stream operator creates the error table before data loading and continue.
- If the structure of the existing error table is not compatible with the error table the Stream operator expects, it terminates the job with an error message before data loading.

**Note:**

A new column, RowInsertTime, was added to the Stream Operator error table for version 13.10. Therefore, you can no longer reuse error tables from prior versions, as valid Stream Operator error tables now have ten columns rather than nine.

If the user requested a queue error table and the max size of the job's schema is 64K, the Stream operator will create the HOSTDATA column as VARBYTE(63666) and the job will continue.

If the user requested a queue error table and the size of the job's schema is or could be greater than 64K, the Stream operator will terminate the job with a TPT16190 error message:

`TPT16190: Error: Row size (128005 bytes) is too big to create queue error table.`

If the HOSTDATA column is defined as VARBYTE(63666) in the operator's error table and the job's schema is or could be greater than 64K, the Stream operator will terminate the job with the TPT16191 error message:

`TPT16191: Error: Row size (128005 bytes) is too big for the error table of which the HOSTDATA column is defined as VARBYTE(63666).`

## Error Limit

Attribute that specifies the approximate number of records that can be stored in the error table before the Stream operator job is terminated.

This number is approximate because the Stream operator sends multiple rows of data at a time to the database. By the time Teradata PT processes the message indicating that the error limit has been exceeded, it may have loaded more records into the error table than the actual number specified in the error limit.

The ErrorLimit specification must be greater than 0. Specifying an invalid value causes the Stream operator job to terminate. By default, the ErrorLimit value is unlimited.

**Note:**

The ErrorLimit specification applies to each instance of the Stream operator.

## DML Statement

### Using Database Macros

The Stream operator uses macros to modify tables rather than actual DML statements. Before beginning a load, the operator creates macros to represent the DML statements. The macros are then iteratively executed in place of the DML statements. The Stream operator also removes the macros after all rows are loaded.

These actions are accomplished through the use of CREATE/DROP MACRO SQL statements.

Use the MacroDatabase attribute to specify the database that contains the macros created by the Stream operator. If not specified, the database used is the database that contains the Restart Log table.

### Using User-Created (Predefined) Macros

For greater efficiency, the Stream operator also supports the use of predefined macros, rather than creating macros from the actual DML statements. A predefined macro is created by the user and resides on the database before a TPT Stream job begins. The macro specifies the type of DML statement (INSERT, UPDATE, DELETE, or UPSERT) being handled by the macro.

When a predefined macro is used, the Stream operator uses this macro directly instead of creating another macro. The use of predefined macros allows the Stream operator to avoid the overhead of creating/dropping macros internally, and also to avoid modifying the data dictionary on the database during the job run. The user needs decide to remove or keep the macros after all rows are loaded. The EXECUTE MACRO privilege is required on the database where the macros are placed.

The Stream operator uses the EXECUTE command to support predefined macros. For more information on using predefined macros, refer to the EXECUTE statement in [Object Definitions and the APPLY Statement](#).

For more information about creating a macro, see *Teradata Vantage™ - SQL Data Definition Language Detailed Topics*, B035-1184.

For more information about executing a macro, see *Teradata Vantage™ - SQL Data Manipulation Language*, B035-1146.

Using predefined macros saves time because the Stream operator does not need to create and drop new macros each time a Stream operator job is run.

The rules for user-created macros are:

- The Stream operator expects the parameter list for any macro to match the list specified by the schema.
- The macro should specify a single prime index operation: INSERT, UPDATE, DELETE, or UPSERT. The Stream operator reports an error if the macro contains more than one supported statement, as in the following example.

Here is an example of the use of predefined macro.

```
/*
/* Description: Use the TPT Stream operator to load data into a      */
/*              table via a predefined macro.                      */
*/
USING CHARACTER SET ASCII
DEFINE JOB LOAD_TABLE_USING_STREAM_OPER
DESCRIPTION 'LOAD TABLE USING STREAM OPERATOR'
(
  DEFINE SCHEMA EMPLOYEE_SCHEMA
  DESCRIPTION 'SAMPLE EMPLOYEE SCHEMA'
  (
    COL1      VARCHAR(5),
    COL2      VARCHAR(5)
  );
  DEFINE OPERATOR DDL_OPERATOR
  DESCRIPTION 'TERADATA PARALLEL TRANSPORTER DDL OPERATOR'
  TYPE DDL
  ATTRIBUTES
  (
    VARCHAR TraceLevel      = 'none',
    VARCHAR PrivateLogName = 'ddloper_log',
    VARCHAR TdpId           = @MyTdpId,
    VARCHAR UserName         = @MyUserName,
    VARCHAR UserPassword     = @MyPassword,
    VARCHAR ErrorList        = '3807'
  );
  DEFINE OPERATOR STREAM_OPERATOR
  DESCRIPTION 'TERADATA PARALLEL TRANSPORTER STREAM OPERATOR'
  TYPE STREAM
  SCHEMA EMPLOYEE_SCHEMA
  ATTRIBUTES
  (
    VARCHAR TraceLevel      = 'none',
    VARCHAR PrivateLogName = 'streamoper_privatelog',
    VARCHAR TdpId           = @MyTdpId,
    VARCHAR UserName         = @MyUserName,
    VARCHAR UserPassword     = @MyPassword,
    VARCHAR ErrorTable       = 'STREAMOPER_ERRTABLE',
    VARCHAR LogTable         = 'STREAMOPER_LOGTABLE',
    VARCHAR PackMaximum      = 'Yes',
  );
)
```

```

INTEGER MaxSessions      = 4,
INTEGER MinSessions      = 4
);

DEFINE OPERATOR FILE_READER
DESCRIPTION 'TERADATA PARALLEL TRANSPORTER DATA CONNECTOR OPERATOR'
TYPE DATACONNECTOR PRODUCER
SCHEMA EMPLOYEE_SCHEMA
ATTRIBUTES
(
  VARCHAR TraceLevel      = 'none',
  VARCHAR PrivateLogName   = 'dataconnoper_privatelog',
  VARCHAR FileName        = 'VARDATAa',
  VARCHAR OpenMode         = 'Read',
  VARCHAR Format           = 'DELIMITED',
  VARCHAR Delimiters       = '|'
);

STEP step1_setup
(
  APPLY
  ('DROP TABLE STREAMOPER_ERRTABLE;'),
  ('DROP TABLE STREAMOPER_LOGTABLE;'),
  ('DROP TABLE STREAMTARGETTABLE'),
  ('CREATE TABLE STREAMTARGETTABLE, Fallback
    (COL1      VARCHAR(5),
     COL2      VARCHAR(5));'),
  ('REPLACE MACRO
    T1INSERT
    (
      COL1 VARCHAR(5),
      COL2 VARCHAR(5)
    ) AS ( INSERT INTO STREAMTARGETTABLE VALUES (:COL1,:COL2);
    );')

  TO OPERATOR (DDL_OPERATOR);
);

STEP step2_load_data
(
  APPLY
  ('EXEC T1INSERT INSERT;')
  TO OPERATOR (STREAM_OPERATOR [2])
)

```

```

SELECT * FROM OPERATOR (FILE_READER [1]);
);
);

```

## Multiple Statement Requests

The Stream operator sends conventional CLIV2 parcels to the database that are immediately applied to the target tables. To improve efficiency, the Stream operator builds multiple statement requests and may also employ a serialize mechanism to help reduce locking overhead.

The most important technique used by the Stream operator to improve performance is the multiple statement request. Placing more statements in a single request is beneficial for two reasons. It reduces:

- Network overhead because large messages are more efficient than small ones.
- Robust mode recovery overhead to one extra restart log table row for each request.

The Stream operator packs multiple statements into a request based upon the Pack attribute specification.

## Enhanced LOB Support

The Stream operator supports enhanced LOB functionality in TPT script mode.

The operator does not support enhanced LOB functionality in TPT API mode.

When the TPT job has one or more LOBs (inline or deferred, regardless of the schema size), the Stream operator can use up to 7 MB message size for sending data to the database. The database has a limit of 7 MB message size for such jobs.

For more information, see [Enhanced LOB Support](#).

## Restrictions and Limitations

This section describes the restrictions and limitations when using the Stream operator.

### Data Types

- The Update, Stream and Inserter operators can all load LOB/JSON/XML in inline mode on all platforms.
  - The Stream operator can load CLOB/BLOB/JSON/XML in inline mode on all non-z/OS platforms. On z/OS, the operator can load CLOB/BLOB/JSON/XML in inline mode if the size is less than or equal to 64,000 bytes. Deferred LOB/XML/JSON is not supported because data sets are not assigned to the LOB, XML, and JSON files.
  - When the job is loading 1 or more deferred LOB/JSON/XML columns, the largest possible pack factor is 4096 because the database can support up to 4096 spool files per request.

- TPT does not support the ST\_Geometry data type in the TPT schema. But, the Stream operator *can* load data to ST\_Geometry columns if the job specifies CLOB or BLOB in the TPT schema.
- UDT data can be loaded in its external type format.

## Error Table

Two jobs loading different tables with different table definition cannot use the same error table.

## Other Notes

- One Stream operator job can load up to 127 database tables or views.
- Only the Teradata SQL INSERT, UPDATE, DELETE, UPSERT, MERGE, and EXECUTE statements are supported. Any other Teradata SQL statements are not supported.
- Data can be loaded into empty or populated database tables.
- By default, duplicate rows are not discarded by the Stream operator for SET tables. Duplicate rows are inserted into the error table. Use the DML option, "IGNORE DUPLICATE ROWS," to override the default if you do not want the duplicate rows to be inserted into the error table.
- Duplicate rows are not discarded by the Stream operator for MULTISET tables; they are inserted into the target tables. The DML options, "MARK DUPLICATE ROWS" and "IGNORE DUPLICATE ROWS" cannot override this behavior. Duplicate rows are allowed for MULTISET tables.
- Row hash level locking is used during the job. The job does not lock the target tables.
- For updates, a record cannot change the value of the primary index of a row, but reflexive updates of other columns are allowed. A reflexive update of a column computes the new value as the result of an expression that involves the current value of one or more columns.
- The atomic UPSERT feature cannot have any of the following syntax or characteristics:
  - INSERT-SELECT
  - UPDATE-WHERE-CURRENT
  - UPDATE-FROM
  - UPDATE-WHERE SUBQUERIES
  - UPDATE-PRIMARY INDEX
  - UPDATE or INSERT that could cause a trigger to be fired
  - UPDATE or INSERT that could cause a join or hash index to be updated
- The target tables cannot be rolled back after the job is complete.
- A DML group cannot use the Array Support feature if the DML group has multiple DML statements. The exception is atomic UPSERT statement. Atomic UPSERT statement can use the Array Support feature.
- A job with multiple DML groups and serialize on cannot use the Array Support feature. A job with a single DML group and serialize on can use the Array Support feature if the DML group contains a single DML statement or an atomic UPSERT statement.

- The Stream operator does not support the TPump's Wildcard INSERT feature.
- The Stream operator does not require a database load slot.

## Job Options

### SERIALIZE

The Serialize option only applies to the Stream operator. Use the Serialize option when correct sequencing of transactions is required. For example, when a job contains a transaction that inserts a row to open a new account, and another transaction updates the balance for the account, then the sequencing of the transactions is critical.

Using the Serialize option in APPLY statements, the Stream operator ensures that operations for a given row occur in the order they are received from the input stream.

To use this option, associate a sequencing key (usually the primary index) with the target table. Each input data row is hashed based on the key to determine the session assigned to process each input row. This allows all rows with the same key to be processed in sequence by the same session, which is especially important if rows are distributed among many sessions.

When using the Serialize option, only one instance of the Stream operator is allowed. Specifying more than one instance causes the Stream operator to terminate with an error.

### SERIALIZE OFF

When the Serialize option is set to OFF, transactions are processed in the order they are encountered, then they are placed in the first available buffer. Buffers are sent to parsing engine (PE) sessions and PEs process the data independently of other PEs. In other words, transactions might occur in any order.

If the Serialize option is not specified, the default is OFF unless the job contains an Upsert operation, which causes Serialize to switch the default to ON.

### SERIALIZE ON

If the Serialize option is set to ON, operations on a row occur serially in the order submitted.

The sequencing key of SERIALIZE ON is specified as one or more column names occurring in the input data SCHEMA definition. These SCHEMA columns are collectively referred to as the key. Usually the key is the primary index of the table being updated, but it can be a different column or set of columns. For example:

```
APPLY
  ('UPDATE emp  SET dept_name = :dept_name
   WHERE empno = :empno;')
```

```
SERIALIZE ON (empno)
TO TARGET_TABLE[1]
```

This APPLY statement guarantees that all data rows with the same key (empno) are applied to the database in the same order received they are received from the producer operator. In this case, the column empno is the primary index of the Emp table.

Note that SERIALIZE ON is local to a specific DML statement. In the following example, a group DML is specified, but only the first statement uses the Serialize option:

```
APPLY
  ( 'UPDATE emp SET dept_num = :dept_num
    WHERE empno = :empno; ')
    SERIALIZE ON (empno)
  ( 'UPDATE dept SET dept_name = :dept_name
    WHERE deptno = :deptno; ')
TO TARGET_TABLE[1]
```

Following are some of the advantages to using the Serialize option, and might improve performance:

- SERIALIZE ON can eliminate the lock delays or potential deadlocks caused by primary index collisions coming from multiple sessions.
- SERIALIZE ON can also reduce deadlocks when rows with non-unique primary index values are processed.

## Robust and Non-Robust Mode

For more robust restartability, use robust mode, which causes every DML operation to be checkpointed and ensures on restart that no operation is applied more than once.

The robust mode requires more writes to a restart log, which might impact performance more, however, using robust mode ensures that a restart avoids reprocessing rows that a normal interval checkpoint might necessitate.

Robust is the default mode for all Stream operator jobs. The Robust attribute turns the mode on or off. If uncertain whether to use robust restart logic, it is always safe to set the Robust parameter to 'Yes'.

- **Robust Mode** – Setting the attribute to "yes" tells the Stream operator to use robust restart logic.

```
VARCHAR Robust = 'Yes' (or 'Y')
```

Robust mode causes a row to be written to the log table each time a buffer successfully completes its updates. Mini-checkpoints are written for each successfully processed row. These mini-checkpoints are deleted from the log when a checkpoint is taken, and are used at restart to identify the rows that have been successfully processed, which permits them to be bypassed at restart. In robust mode, each row is processed only once. The larger the Pack factor, the less overhead is involved in this activity.

Choosing the Robust mode is particularly useful to avoid problems with data integrity and unacceptable performance. Robust mode is recommended in the following situations to avoid having an adverse affect on restarts:

- **INSERTs into multi-set tables** – Robust mode prevents the insertion of duplicate rows, which could insert the same row a second time.
- **UPDATEs based on calculations** – Robust mode prevents the duplicate application of calculations.
- **Large Pack factors** – Robust mode does not involve the application and rejection of duplicate rows after restarts, which is a time-consuming process of logging errors to the error table.
- **Time-stamped data** – Robust mode prevents the possibility of stamping identical rows with different time stamps, resulting in duplicate rows.

If rows are reapplied in non-robust mode, each reapplied row is marked with a time stamp that is different from the original row even though all of the other data is identical. To the database, these reapplied rows are different rows with the same primary index value, so they are inserted even though they are duplicates.

- **Non-Robust Mode** – Setting the attribute to “no” tells the Stream operator to use simple restart logic rather than robust logic.

```
VARCHAR Robust = 'No' (or 'N')
```

In a non-robust mode, restarts begin where the last checkpoint occurs in a job. Because some additional processing will most likely take place after the checkpoint is written, the requests that occur after the checkpoint are resubmitted by the Stream operator as part of the restart process. For Deletes, Inserts and Upserts, this does not usually cause a problem or harm the database; however, re-running statements generates more rows in the error table because the operator will be attempting to insert rows that already exist and to delete rows that do not exist.

Re-attempting updates can also be a problem if update calculation, for example, is based on existing data in the row, such as adding 10% to an amount. Doing the update calculation a second time add an additional 10% to the amount, thus compromising data integrity. In this type of update, it is best to use robust mode to ensure that no DML operation is applied more than once.

The non-robust (or simple restart) method does not involve the extra overhead that comes with the additional inserts to the restart log table that are needed for robust logic, so overall processing is notably faster.

## Recovery Logic and Overhead

In Robust mode, the Stream operator writes one row in the restart log table for each request issued. This collection of rows in the restart log table can be referred to as the request log. Because a request is guaranteed by the database to either completely finish or completely roll back, the request log will always accurately reflect the completion status of a load operation. Thus, the request log overhead for restart logic decreases as the number of statements packed per request increases. During the checkpoint process, the Stream operator flushes all pending changes from internal storage to the database and also deletes the

request log rows. The larger the checkpoint interval, the larger the request log is going to grow. In the event of a Robust mode restart, the Stream operator will use the request log to avoid the erroneous reapplication of database changes.

The Stream operator, in simple (non-robust) mode, provides basic checkpoints. If a restart occurs, then some requests will likely be reprocessed. This is adequate protection under some circumstances. Simple logic is adequate in certain DML statements that can be repeated without changing the results of the operation.

Examples of statements that are *not* simple include the following:

- Inserts into tables that allow duplicate rows (MULTISET tables).
- Self-referencing DML statements like: "UPDATE FOO SET A=A+1...", or "UPDATE FOO SET A = 3 WHERE A=4"

## Data Quality Affects Performance

It is more important to have error-free data when using the Stream operator than with other Teradata PT operators. If data contains errors, a large Pack factor can slow performance because of the way Stream handles errors.

For example, if the statement independence feature is not used and several hundred statements are packed, the entire request (when an error occurs) is rolled back. The Stream operator then removes the error-producing statement and reissues the entire request. Such a process can be costly from a performance standpoint.

If the statement independence feature is used and several hundred statements are packed, the entire request (when an error occurs) will not be rolled back or reissued. The error will be inserted into the Stream operator's error table if the errors are marked. The statement independence feature can improve performance.

Statement independence feature only supports the INSERT statement. No other DML statements are supported.

## Statement Packing

To provide optimal performance, the Stream operator packs individual DML statements into a larger multistatement request based on the rate specified by the Pack attribute. This type of processing requires less overhead than multiple individual requests.

The Stream operator submits these multistatement requests using macros which it creates to hold the requests. The macros are then executed instead of running each individual DML statement.

The macros are automatically removed after the job is complete. The use of macros in place of lengthy requests helps to minimize both network and parsing overhead.

## Specifying the Pack Rate

The Pack attribute specifies the number of statements in a multistatement request. Specifying a Pack rate improves network/mainframe efficiency by reducing the number of sends and receives between Teradata PT and the database. A maximum of 2400 statements can be specified.

Trial and error might be required to determine the best Pack rate for a Stream job. As the Pack rate is increased, the throughput improvement is usually great at first, then falls off. In other words, going from a Pack rate of 1 to 2 could provide huge performance gains, and going from 2 to 4 could be just as beneficial, but moving from 8 to 16 might cause a performance drop.

If the PackMaximum attribute is set to 'Yes', the Stream operator determines the maximum pack for the job, and then reports it.

Two factors to consider are:

- The maximum Pack factor based on Stream operator restrictions
- The optimal Pack factor for a particular job

These two factors might not be equal. The maximum rate lets you know the upper limit, but performance might improve at a smaller rate. For this reason, it is recommended that PACKMAXIMUM not be used for production jobs until you determine the optimal Pack factor.

## Tuning the Pack Factor

Packing multiple statement requests improves network/mainframe efficiency by reducing the number of sends and receives between the application and the database.

To determine the ideal pack factor to specify in the Pack attribute, first use the PackMaximum attribute by setting it to 'Yes'. Setting this attribute to Yes on the first job run sets up iterative interactions with the database to heuristically determine the maximum possible pack factor. At the end of the run, this value is displayed in the Stream operator's logged output. Specify that determined value in the Pack attribute on subsequent runs. Set the PackMaximum to 'No'.

Alternatively, the Stream Driver TD\_Evt\_PackFactor event returns the current pack factor when queried. This value is available after a connection is initiated and before it is terminated.

The Stream operator will fill up to the max Packing factor or until the buffer is filled on a request-by-request basis when the following conditions are met:

- The schema has variable-length columns and
- Array Support is on and
- One of the following:
  - The PACK factor is set to 2400
  - The PACKMAXIMUM is set to 'Y[es]'
  - Neither the PACK factor attribute nor PACKMAXIMUM attribute is populated

The user will be informed the "floating" Pack factor via a new informational message similar to the following:

```
**** 14:50:34 The PACK factor has changed. The minimum PACK factor is about
1270 data records per request. The maximum PACK factor is about 1298 data
records per request.
```

In the message, the word "about" indicates that worker instances can have a different Pack factor.

## Array Support

The Array Support feature allows DML requests containing only a single statement to be executed once for each of multiple rows of input data, each row specified being one of the members of the array. The DML must be contained within an APPLY statement that includes Stream operator and does not set ArraySupport to Off.

Use of array support improves performance by:

- Increasing the amount of data that can be sent per request from a total of approximately 64 KB to a total of approximately 1 MB, with a limit of 64 KB per input data row.
- Improvements in internal database processing as a result of using such requests.

## Latency Interval

Latency is the interval value, expressed in seconds, between the flushing of stale buffers. Latency interval is an option that is exclusively used by the Stream operator.

In normal operations (without latency), the Stream operator reads data from the data stream until its buffer is full, then it writes all buffered rows to the database. The data is written to the database only when the buffer is full or when a checkpoint is taken. However, a latency interval (for example, set to 5400 seconds) causes the following:

- The Stream operator reads data from the data stream, and empties its buffer, writing the contents to the database every 90 minutes (5400 seconds) regardless of whether it is full.
- If the buffer fills up within the time period (in this case, 90 minutes), it writes to the database as it would during normal operation.

To set the latency interval, use the following syntax:

```
tbuild -l <LatencyInterval> -f <filename>
```

The value used for the latency interval must be a non-zero unsigned integer. The guiding factor is how stale you are willing to allow data to be.

For example, to run a continual load script with a latency interval of two hours, enter:

```
tbuild -l 7200 -f continualload
```

## MacroCharSet

The MacroCharSet option only applies to the Stream operator.

The MacroCharSet option specifies the server storage character set name for the character field in the schema. The Stream operator uses the server storage character set name to generate a CHARACTER SET clause for each character field when creating the macros. The Stream operator creates a macro for each DML statement specified in the job. The correct server storage character set is required to avoid character set translation errors.

There are two ways to define the server storage character set name:

- Specify the field-level MACROCHARSET (<field-server-character-set>) clause after the declaration of a character column in the DEFINE SCHEMA statement.

If a field-level MACROCHARSET (<field-server-character-set>) clause is specified for a given character field in the schema, the field server character set will be used for the given character field.

- Specify the global-level DEFAULT MACROCHARSET (<global-server-character-set>) clause in the DEFINE SCHEMA statement.

If the global-level DEFAULT MACROCHARSET (<global-server-character-set>) clause is specified, the global server character set will be used for each character field that has no field-level MACROCHARSET clause in the schema.

If the global-level DEFAULT MACROCHARSET (<global-server-character-set>) is not specified, the server storage character set will be LATIN or UNICODE (depending on the client session character set) for each character field that has no field-level MACROCHARSET clause in the schema.

It will be LATIN if the client session character set name:

- Is ASCII
- Is EBCDIC
- Ends in \_0A or \_0E
- Ends in \_zx, where "z" is other than 0 (digit zero)
- Does not end in \_zx or \_zxx, except for KATAKANAEBCDIC, UTF-8, or UTF-16

It will be UNICODE in all other client session character sets.

When you are loading to character columns with different server storage character sets, it is recommended to use the field-level MACROCHARSET (<field-server-character-set>) clause. For example, you are loading data to a target table using a Japanese client session character set with the following table definition: EMP\_ID INTEGER, EMP\_NAME VARCHAR(40) CHARACTER SET UNICODE, EMP\_SEX CHAR(1) CHARACTER SET LATIN.

To specify the field-level MACROCHARSET clause:

```
DEFINE SCHEMA EMP_SCHEMA
(
    EMP_ID    INTEGER,
```

```

EMP_NAME VARCHAR(40) MACROCHARSET (UNICODE),
EMP_SEX CHAR(1) MACROCHARSET (LATIN)
);

```

The job will terminate with an error if you specify:

- A field-level MACROCHARSET (<field-server-character-set>) clause on a non-character column.
- An invalid server storage character set. The job will terminate with "RDBMS error 3706: Syntax error: Expected Character Data Type."

For the server storage character set names, see *Teradata Vantage™ - Analytics Database International Character Set Support*, B035-1125.

## Operational Considerations

### NoPI Tables

Operations other than Insert on target tables defined as NoPI exhibit poor performance, unless there is an appropriate secondary index path to the data rows.

### Space Requirements

Always estimate the final size of the Stream target table, and make sure that the destination database has enough space to accommodate the Stream job.

If the database that owns the Stream target table, log table, or error table runs out of space, the database returns an error message and the Stream operator terminates the Stream operator job. When this happens, you must allocate more space to the database before you can restart the job.

### Sessions and Instances

Because the Stream operator uses Teradata SQL sessions in its communication with the database, it does not use database load slots. The Stream operator provides continuous updates using row level locking, allowing constant load operations in the background during normal system use.

Both a minimum and a maximum number of sessions can be used by the Stream operator. The minimum specification is one. The default is one session for each operator instance.

The number of sessions is evenly distributed among the number of operator instances. If 20 sessions are requested and four instances of the operator are invoked, then each instance will receive five of the sessions.

### Other Session Limits

The values that you specify with the Stream operator MinSessions and MaxSessions attributes are not the only factors that limit the number of sessions that the Stream operator establishes with the database. The other limiting factors are:

- The platform limit on the maximum number of sessions per application.
  - On workstation-attached client systems for UNIX, Linux, and Windows systems, this value is defined in the COP Interface software file, `clispb.dat`, under the `max_num_sess` variable.
  - On mainframe-attached z/OS client systems, this value is defined in the HSHSPB parameter under the IBCSMAX setting.
  - On mainframe-attached z/OS client system, use the TDP SET MAXSESSIONS command to specify a platform limit.
- The limit of the network protocol software on workstation-attached systems.

When the Stream operator executes, the actual session limit is determined by the first limiting factor that is encountered.

## Checkpointing and Restarting

Checkpoint options control how often a row is written to the checkpoint file for the purposes of restarting a job. Unless otherwise specified, a checkpoint is taken at the start of and at the end of the input data. Since this process does not provide granular restartability in the case of longer running jobs, checkpoint intervals can be user-specified in terms of seconds using the command line option `-z`.

For example, the following command indicates that a checkpoint will be taken every 120 seconds:

```
tbuild -f <script file name> -z 120
```

## Dropped Tables

Some tables are created by the database during the execution of a job, and others must be user-created before the job begins. The log table and error table are automatically created by Teradata PT when you run the job script. The error table is dropped during the Cleanup phase if no errors were detected during the job, unless the DropErrorTable attribute is set to No. The log table is dropped after the job completes successfully.

If a job terminates abnormally, then the log and error tables are not dropped. If you want to restart the job from scratch, manually drop these tables by running a BTEQ script.



### NOTICE

Care must be taken dropping the target tables manually using a BTEQ script. If something goes wrong with a Stream operator job, and you drop the target table manually and then try to rerun the job, original data can be lost.

## DML Option Categories

The DML Options can be classified into four logical categories, which can be specified in any order:

- ARRAY SUPPORT

- SERIALIZE
- USE
- DML ERROR OPTIONS, which include:
  - MARK
  - IGNORE
  - INSERT FOR

The following restrictions apply:

- Multiple ARRAY SUPPORT Options cannot be specified.
- Multiple SERIALIZE Options cannot be specified.
- Multiple USE Options cannot be specified.
- Multiple DML Error Options (MARK,IGNORE, INSERT FOR) can be specified in one block. However these cannot be interleaved with the other three DML Option categories.

## Temporal Tables

When using the Stream operator to load data into temporal tables, the temporal qualifiers, such as CURRENT VALIDTIME and SEQUENCED VALIDTIME, must precede the DML statements, as in the following example.

```
APPLY
('SEQUENCED VALIDTIME INSERT INTO <target table> ... ;')
TO OPERATOR (STREAM_OPERATOR)
SELECT * FROM OPERATOR (<producer operator>);
```

When using an atomic Upsert, the temporal qualifier must precede the UPDATE statement, not the INSERT statement, as in the following example.

```
APPLY
('SEQUENCED VALIDTIME UPDATE <target table> SET ... WHERE ... ;',
 'INSERT INTO <target table> ... ;')
INSERT FOR MISSING UPDATE ROWS
TO OPERATOR (STREAM_OPERATOR)
SELECT * FROM OPERATOR (<producer operator>);
```

# Update Operator

## Update Operator Capabilities

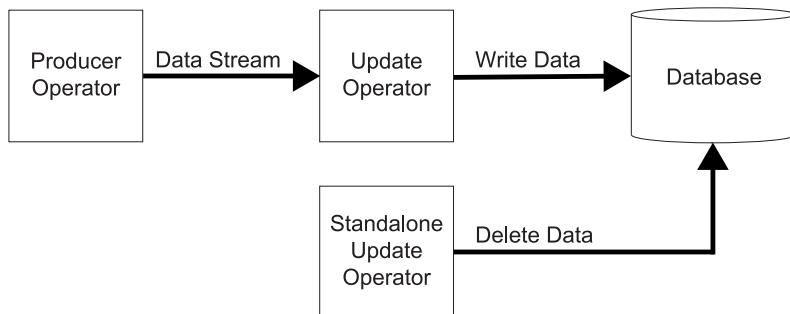
The Update operator, a consumer operator, emulates the Teradata MultiLoad utility to perform highly scalable and parallel Inserts, Updates, Deletes, and Upserts into new or preexisting database tables in a single pass. The Update operator can also be used to insert new rows into the database, but it cannot perform Select operations.

## Update Operator as Standalone Operator

The Update operator can also function as a standalone operator when it is executing the Delete task and if no data is required. The Delete Task permits the high-speed deletion of table rows based on a non-index value.

For information on the Delete task option, see [Delete Task Option](#).

The following figure illustrates how the Update operator works.



## The Update Operator Function in a Teradata PT Job Script

When you use the Update operator in a Teradata PT job, each parallel instance of the Update operator:

1. Log on to the database, using your user name, password, database name, and account ID information specified in the job script.
2. Loads the data from the Teradata PT data streams into the database.
3. Logs off from the database.
4. If the Update operator job is successful, terminates the job and provides information about the job in the log or to the user, such as:
  - Total number of records sent to the database

- Number of errors posted to the error tables
  - Number of Inserts, Updates, and Deletes applied
  - A status code indicating the success of the job
5. If the job is unsuccessful, terminate the job and provide information about the job so that you can correct any problem and restart the job.

## Functional Description of the Update Operator

During an Update job, the Update operator loads database tables. Each instance retrieves rows from the Teradata PT data streams and sends them to the database. Multiple parallel instances can be used to improve the performance of the update.

The Update operator reads a data block only once. The operator reads rows from a data stream, writes that data to a buffer, then sends that buffer of data across sessions to the available AMPs.

This process is dependent on all changes to tables being keyed on the primary index. Thus, all transactions must be primary index operations when using the Update operator. An exception to this rule is the use of the Delete Task. For more information, see [Delete Task Option](#).

## Update Phases

Update operations have two phases:

- **Acquisition Phase** – Data from the input stream is transmitted to the AMPs, and access locks are placed on the target tables, limiting table access to read-only. The acquisition phase is complete when all data rows are on the appropriate AMPs where their changes will be applied.

Records are sorted by the hash value of the primary index value. This sorting order becomes the sequence in which they are applied to a target table. Sorted records are placed in temporary work tables that require permanent space for the duration of the job.

One work table per target table is created by the database. Because the acquisition phase involves writing only to work tables, target tables are left available for user access.

### Note:

When the job has multiple work tables and uses the Extended MultiLoad Protocol, only the first work table is created. The Update operator inserts records into the first work table during the acquisition phase.

- **Application Phase** – Sorted input records are applied to data blocks of target tables using the appropriate DML commands (insert, update, delete). Each target block is read once into memory, and all changes are applied at that time. The access lock on target tables is upgraded to a write lock, so tables are not available for user access until the phase is complete. At the end of this phase, the work tables are dropped during a subsequent clean-up phase.

## MultiLoad Utility and the Update Operator

The following table lists the operating features and capabilities of the Teradata MultiLoad utility and indicates whether they are supported by the Update operator or another Teradata PT operator.

**Update Operator Feature Support**

Multiload Utility Feature	Update Operator
Absolute field positioning (handled by the FIELD command)	Not supported
Access Modules	Supported, using the DataConnector operator
ANSI Date	Supported
Checkpoint/Restart	Supported
Character Sets	Supported, using the USING CHARACTER SET clause before the DEFINE JOB statement
Configuration File	Supported, using the <b>tbuild</b> command line job attribute option (-v)
CREATE TABLE Statement	Supported, using the DDL operator
DATABASE Statement	Supported
DELETE Statement	Supported, using the DeleteTask attribute and the APPLY statement
DROP TABLE Statement	Supported, using the DDL operator
Environment variable	Not supported
Error Limit	Supported
IF-ELSE-ENDIF constructs	Not supported
Indicator Mode	Supported, using the DataConnector operator
INMOD Routines	Supported, via the MultiLoad INMOD Adapter operator
Maximum/Minimum Sessions	Supported
Nonindicator Mode	Supported, using the DataConnector operator
Notify	Supported
NULLIF Clauses	Supported, but not the XB, XC, and XG data types
Operating System Command	Supported, using the OS Command operator
Pause Acquisition	Supported
Predefined system variables (i.e., SYSDATE, SYSDAY)	Not supported
Record Formats	Supported, using the DataConnector operator

Multiload Utility Feature	Update Operator
RECORD <i>n</i> THRU <i>m</i>	Supported in a limited form by the DataConnector operator, allowing you to read in the first " <i>m</i> " rows of a file, effectively allowing "RECORD 1 THRU <i>m</i> "
Replication Services override	Supported
Routing of messages to alternate file	Supported via Logger Services
RUN FILE	Supported via Teradata PT script language
Schemas, more than one	Not supported
Show Version information	Supported
SQL statements (such as CREATE TABLE, DROP TABLE, etc.)	Supported, using the DDL operator
Tenacity	Supported
User-defined variables	Limited support via script language
Wildcard INSERT	Not supported (cannot use "INSERT INTO tablename.*,")

## Syntax

Use the following specifications and attributes to define the Update operator in a Teradata PT job script.

## Required Specifications

The following specifications are required to define an Update job.

### Required Syntax for the Update Operator

Specification	Description
TYPE	<p>Operator type. Always UPDATE for consumer or standalone operator.</p> <p>As a consumer operator, which requires a SELECT statement as part of the APPLY statement:</p> <pre>APPLY 'DELETE FROM TABLE table_a WHERE col1 &lt; :col1;' TO OPERATOR (UPDATE_OPERATOR[1]) SELECT * FROM OPERATOR (READ_OPERATOR[1]);</pre> <p>As a standalone operator:</p> <pre>APPLY 'DELETE FROM TABLE table_a;' TO OPERATOR (UPDATE_OPERATOR[1]);</pre> <p>The standalone operator <i>must not</i> contain a SELECT statement.</p>

Specification	Description
UserName	Operator attribute providing the name of the user for the Update operator logon sessions.
UserPassword	Operator attribute providing the password associated with the user name.
TargetTable	Operator attribute providing the names of the target tables.

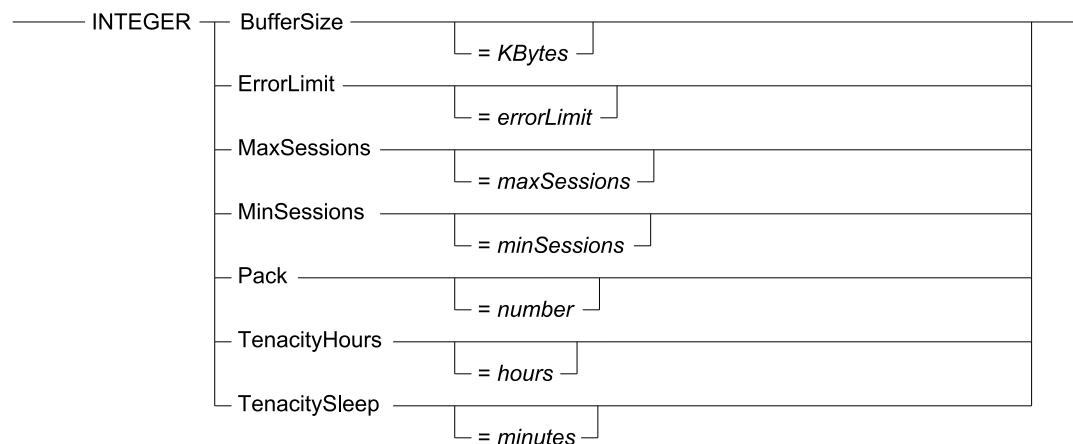
## Required and Optional Attributes

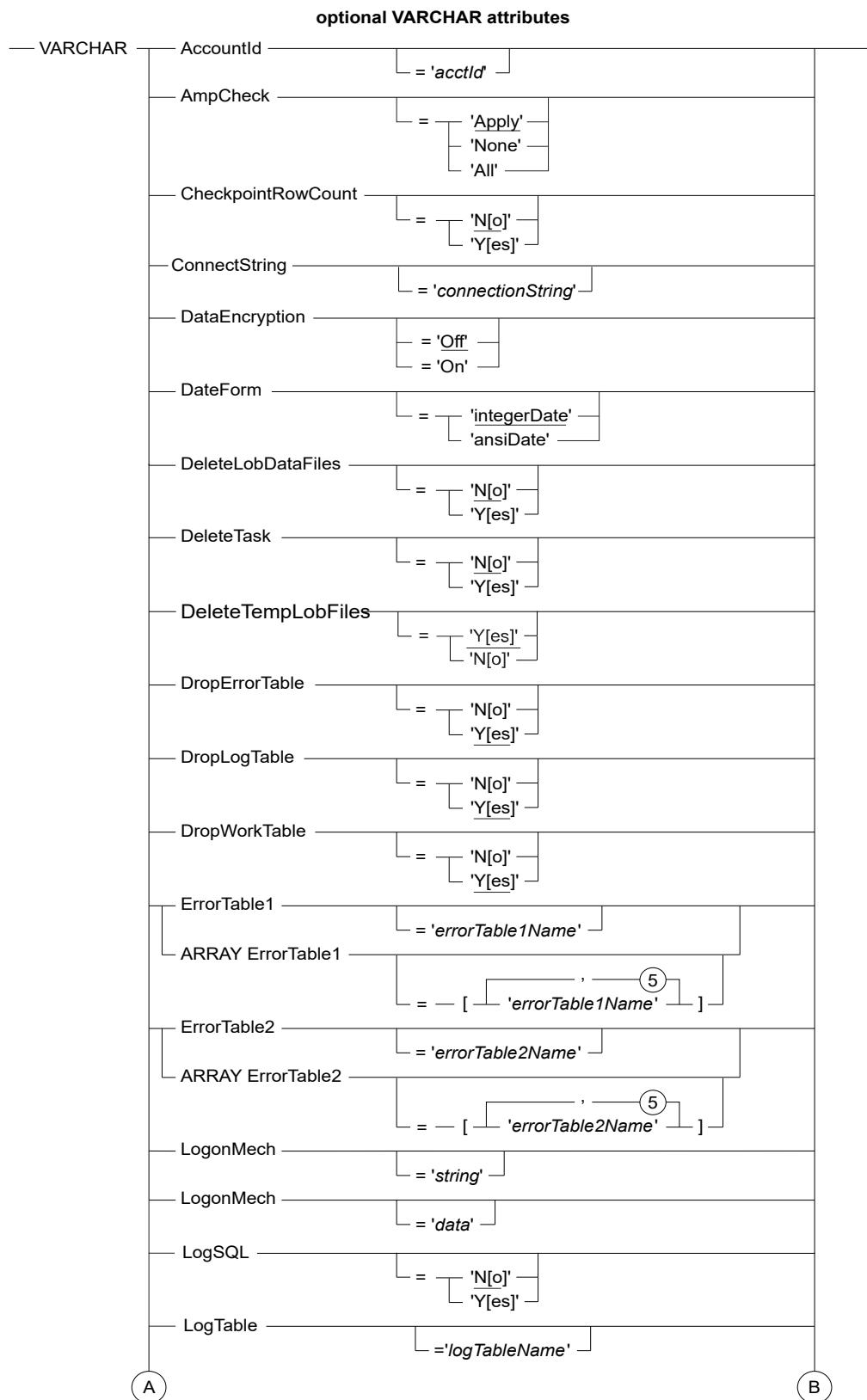
Use the attribute definition list syntax in the Teradata PT DEFINE OPERATOR statement to declare the required and optional attribute values for the Update operator.

optional INTEGER attributes

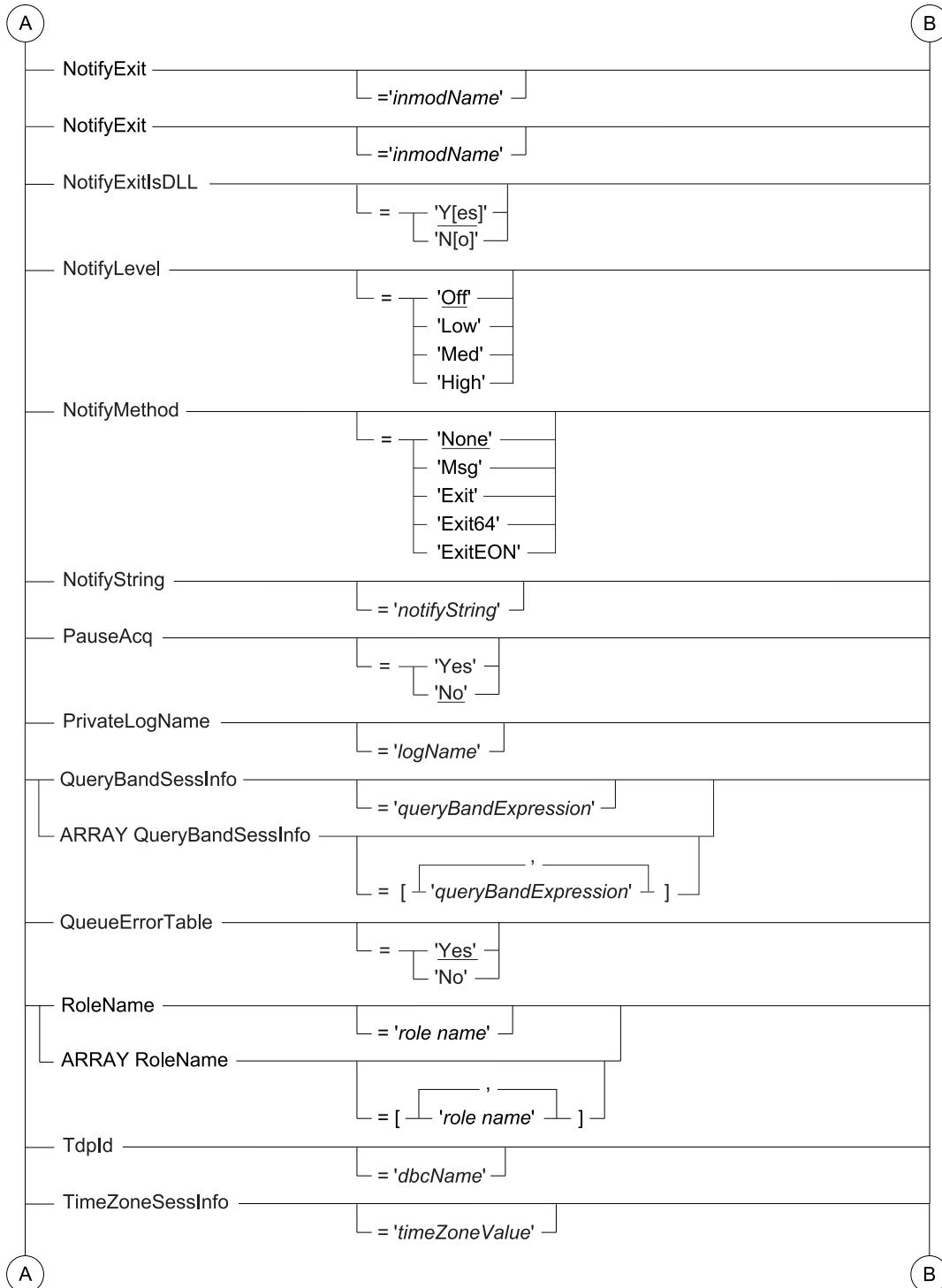


optional INTEGER attributes

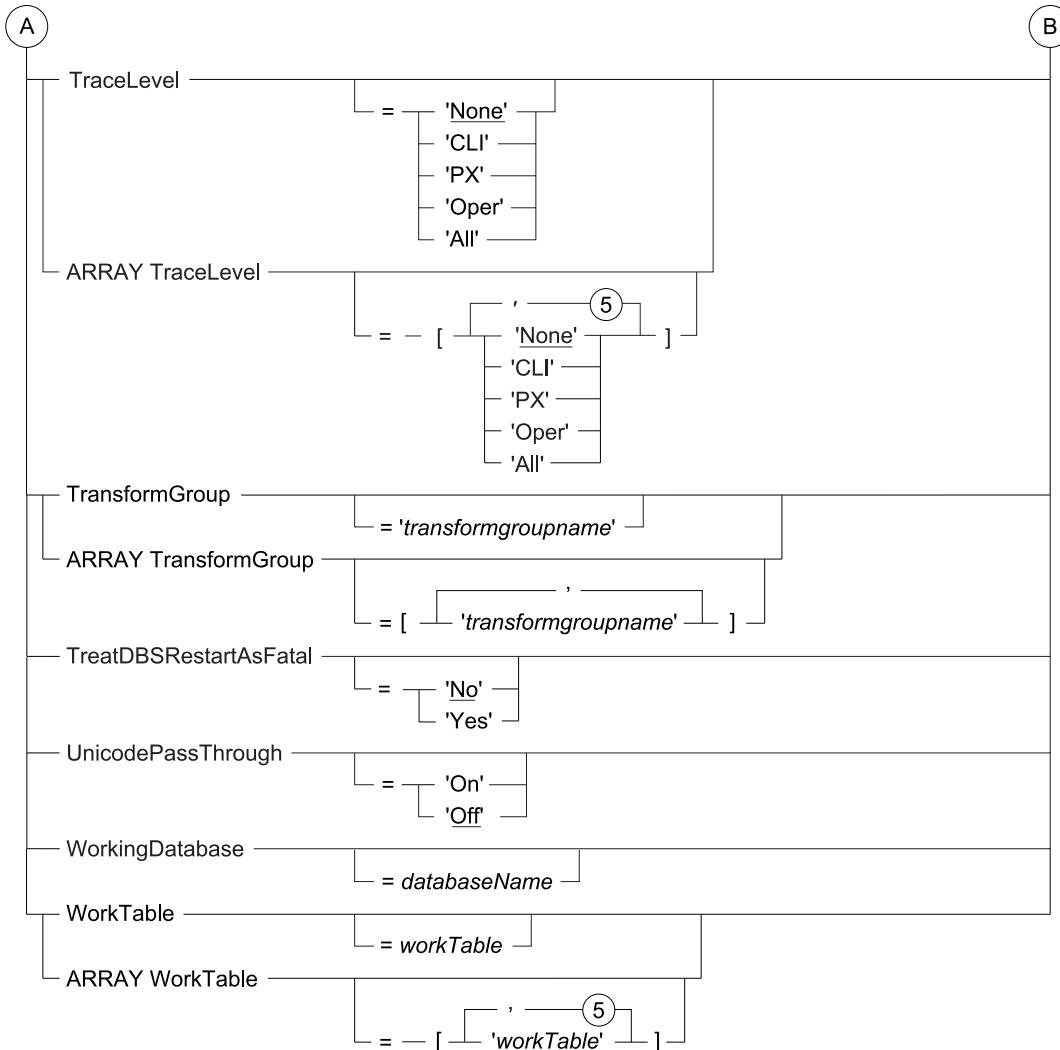




## optional VARCHAR attributes (continued)



## optional VARCHAR attributes (continued)



where:

Syntax Element	Description
AccountId = 'acctId'	Optional attribute that specifies the account associated with the user name. If omitted, it defaults to the account identifier of the immediate owner database.
AmpCheck = 'option'	Optional attribute that specifies the Update operator response to a down AMP condition: <ul style="list-style-type: none"> <li>'None' = allows the Update job to start, restart, or continue as long as no more than one AMP is down in a cluster.</li> <li>'Apply' = inhibits the Update operator job from entering or exiting the application phase when an AMP is down (default).</li> <li>'All' = pauses the Update operator job when an AMP is down.</li> </ul>

Syntax Element	Description
	<p><b>Note:</b></p> <p>All of the target tables in the Update operator job must be fallback tables for the job to start, restart, or continue with a down AMP. The job does not start or restart if any of the target tables are nonfallback.</p>
BufferSize = <i>KBytes</i>	<p>Optional attribute that specifies the output buffer size, in kilobytes, that is used for sending Update parcels to the database.</p> <p>The output buffer size and the size of the rows in the Update table determine the maximum number of rows that can be included in each parcels to the database. A larger buffer size reduces processing overhead by including more data in each parcels.</p> <p>Allowable values are 1 through 16384, but if you specify a value of 16384, the actual buffer size gets set to 16775552 bytes, which is less than the full 16MB. If the value less than 1, an error message results and job terminates.</p> <p>The default buffer size is 1024K bytes when the operator is talking to a Teradata Database 16.00 or later.</p> <p>The default buffer size is 64K bytes when the operator is talking to a pre-16.00 Teradata Database version.</p> <p>The maximum allowed buffer size is usually 16384 Kbytes. Values are evaluated when the connection to the database is made.</p> <p>If the supplied buffer size is too large, the operator scales it back to the maximum allowed buffer size.</p>
CheckpointRowCount - ' <i>option</i> '	<p>Optional attribute that tells the Update operator to enable or disable outputting the rows sent at checkpoints.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>• 'No' ('N') = The Update operator will not output rows sent at checkpoints (default);</li> <li>• 'Yes' ('Y') = The Update operator will output rows sent at checkpoints</li> </ul> <p>This attribute is only available in the TPT script mode.</p>
ConnectionString = ' <i>connectionString</i> '	<p>Optional attribute that specifies the connection string. The connection string will be passed to CLI. CLI will validate the connection string.</p> <p>For information on connection string, see <i>Teradata® Call-Level Interface Version 2 Reference for Workstation-Attached Systems</i>, B035-2418.</p> <p><b>Note:</b></p> <p>The TPT Connection String feature is available on all platforms, except z/OS.</p>
DataEncryption = ' <i>option</i> '	<p>Optional attribute that enables full security encryption of SQL requests, responses, and data.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>• 'On' = all SQL requests, responses, and data are encrypted.</li> <li>• 'Off' = no encryption occurs (default).</li> </ul>
DateForm = ' <i>option</i> '	<p>Optional attribute that specifies the DATE data type for the Update operator job, where:</p> <ul style="list-style-type: none"> <li>• 'integerDate' = the integer DATE data type (default)</li> <li>• 'ansiDate' = the ANSI fixed-length CHAR(10) DATE data type</li> </ul>

Syntax Element	Description
DeleteLobDataFiles = ' <i>option</i> '	<p>Optional attribute that specifies whether to delete deferred LOB data files from the Data Connector Producer once the rows are committed to the database. Valid values for '<i>option</i>' are:</p> <ul style="list-style-type: none"> <li>• 'Yes' (or 'Y') = Delete deferred mode LOB data files once rows are committed to the database</li> <li>• 'No' (or 'N') = do not delete deferred mode LOB data files (default). Specifying any other value results in an error.</li> </ul>
DeleteTask = ' <i>option</i> '	<p>Optional attribute that specifies whether to perform the delete task that deletes data from a single database table. The Delete Task deletes rows much more quickly than a plain DELETE SQL statement. You cannot use a delete task on a view.</p> <p>The valid values for <i>option</i> are:</p> <ul style="list-style-type: none"> <li>• 'Yes' (or 'Y') = perform the delete task. The delete task will know whether to retrieve a row based on whether the APPLY statement has a SELECT section in it.</li> <li>• 'No' (or 'N') = do not perform the delete task (default).</li> </ul> <p>Specifying any other value results in an error.</p> <p>The absence of any value is the same as a 'No' value, that is, the Update operator will execute an IMPORT task, and none of these rules apply.</p> <p>If the DeleteTask attribute processing is enabled, the only other optional attributes with any importance are:</p> <ul style="list-style-type: none"> <li>• TenacityHours</li> <li>• TenacitySleep</li> <li>• AmpCheck</li> </ul>
DeleteTempLobFiles	<p>Optional attribute that tells the operator whether or not to delete the temporary LOB directory and the temporary LOB files in the directory at cleanup.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>• 'Y[es]' = Tells the operator to delete the temporary LOB directory at cleanup. This is the default.</li> <li>• 'N[o]' = Tells the operator not to delete the temporary LOB directory at cleanup. The user is responsible for deleting the temporary LOB directory.</li> </ul> <p><b>Note:</b></p> <p>When the temporary LOB files exist in a remote NFS mount directory, the suggestion is to set this attribute to 'No' and the operator will not spend time deleting the temporary LOB files.</p> <p><b>Note:</b></p> <p>Temporary LOB files will be created in the temporary LOB directory.</p> <p>The naming convention for the temporary LOB directory will be as follows:</p> <ul style="list-style-type: none"> <li>• When the producer's LobDirectoryPath attribute is set, the name of the temporary LOB directory will be this: &lt;LobDirectoryPath&gt;/&lt;job id&gt;_s&lt;job step number&gt;_TempLOBDir</li> <li>• When the producer's LobDirectoryPath attribute is not set, the name of the temporary LOB directory will be this: &lt;current working dir&gt;/&lt;job id&gt;_s&lt;job step number&gt;_TempLOBDir</li> </ul> <p>This attribute is not supported in TPTAPI.</p> <p>This attribute is not supported on z/OS.</p>

Syntax Element	Description
DropErrorTable = ' <i>option</i> '	<p>Optional attribute that specifies whether or not error tables are dropped upon successful completion of the update job, even if the error tables are empty.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>‘Yes’ = Update operator will drop the error table only if empty (default). Teradata PT automatically executes a DROP TABLE statement.</li> <li>If you run many update jobs on a regular basis, checking ‘Yes’ would cause lots of updates to the Data Dictionary, resulting in performance problems.</li> <li>‘No’ = Update operation will not drop the error table, even if empty. The user must manually execute a DROP TABLE statement on the error table.</li> <li>If the tables are not dropped prior to running a Teradata PT job that would use tables by the same name, running that job may result in an error or in unpredictable results.</li> </ul>
DropLogTable = ' <i>option</i> '	<p>Optional attribute that specifies whether or not restart log table is dropped upon successful complete of the load job.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>‘Yes’ = Update operator will drop the restart log table (default). Teradata PT automatically executes a DROP TABLE statement.</li> <li>If you run many update jobs on a regular basis, checking ‘Yes’ would cause lots of updates to the Data Dictionary, resulting in performance problems.</li> <li>‘No’ = Update operation will not drop the restart log table. The user must manually execute a DROP TABLE statement on the restart log table.</li> <li>If the table is not dropped prior to running a Teradata PT job that would use the table by the same name, running that job may result in an error or in unpredictable results.</li> </ul>
DropWorkTable = ' <i>option</i> '	<p>Optional attribute that specifies whether or not the work tables are dropped upon successful completion of the load job.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>‘Yes’ = Update operator will drop the work table (default). Teradata PT automatically executes a DROP TABLE statement.</li> <li>If you run many update jobs on a regular basis, checking ‘Yes’ would cause lots of updates to the Data Dictionary, resulting in performance problems.</li> <li>‘No’ = Update operation will not drop the work table. The user must manually execute a DROP TABLE statement on the work table.</li> <li>If the tables are not dropped prior to running a Teradata PT job that would use tables by the same name, running that job may result in an error or in unpredictable results.</li> </ul>
ErrorLimit = <i>limit</i>	<p>Optional attribute that specifies the approximate number of records that can be stored in one of the error tables before the Update operator job is terminated.</p> <p>This number is approximate because the Update operator sends multiple rows of data at a time to the database. By the time the Update operator processes the message indicating that the error limit has been exceeded, it may have loaded more records into the error table than the actual number specified in the error limit.</p> <p>The ErrorLimit specification must be greater than 0. Specifying an invalid value will cause the Update operator job to terminate. By default, ErrorLimit value is unlimited.</p> <p>The ErrorLimit specification applies to each instance of the Update operator.</p>

Syntax Element	Description
ErrorTable1 = 'errorTableName'	<p>Optional attribute that specifies the name of the first error table, called the acquisition error table. This table contains information about data errors that occur during the acquisition phase of the Update operator job.</p> <p>This must be a new table name. You cannot use a name that duplicates the name of an existing table unless you are restarting a paused Update operator job.</p> <p><b>Note:</b></p> <p>If the name is not supplied, it is created by the Update operator. The name of the created table is prepended with an identifying ttname_ET, thus ensuring uniqueness (for example ttname_ET if VARCHAR is used, or ttname1_ET, ttname2_ET, ..., ttname5_ET, if VARCHAR ARRAY is used).</p> <p>For more information, see <a href="#">ErrorTable</a> and <a href="#">Auto-Generation of Error and Work Tables</a>.</p>
ErrorTable2 = 'errorTable2Name'	<p>Optional attribute that specifies the name of the second error table, called the application error table. This table contains information about data errors that occur during the application phase of the Update operator job.</p> <p>This must be a new table name. Do not use a name that duplicates the name of an existing table unless you are restarting an Update operator job.</p> <p><b>Note:</b></p> <p>If the name is not supplied, it is created by the Update operator. The name of the created table is prepended with an identifying ttname_UV, thus ensuring uniqueness (for example ttname_UV if VARCHAR is used, or ttname1_UV, ttname2_UV, ..., ttname5_UV, if VARCHAR ARRAY is used).</p> <p>For more information, see <a href="#">ErrorTable</a> and <a href="#">Auto-Generation of Error and Work Tables</a>.</p>
LogonMech = 'string'	<p>Optional attribute that specifies which logon mechanism to use.</p> <p><b>Note:</b></p> <p>Specification of this attribute may be required for some authentication methods.</p> <p>The job terminates if the attribute exceeds 8 bytes.</p> <p>For information on specification requirements for LogonMech, see “Logon Security” in <i>Teradata® Parallel Transporter User Guide</i>, B035-2445.</p>
LogonMechData = 'data'	<p>Optional attribute that passes along additional logon data.</p> <p><b>Note:</b></p> <p>Specification of this attribute is required for some external authentication methods. For information on specification requirements for LogonMechData, see “Logon Security” in <i>Teradata® Parallel Transporter User Guide</i>, B035-2445.</p>
LogSQL = 'option'	<p>Optional attribute that controls how much of the job's SQL to enter into the log.</p> <p>Valid options are:</p> <ul style="list-style-type: none"> <li>• 'Yes' = Output the full SQL to the log. The maximum length is 1M.</li> <li>• 'No' = Do not output SQL to the log.</li> </ul>

Syntax Element	Description
	<ul style="list-style-type: none"> <li>No value or attribute omitted = accept the pre-defined limit, which displays up to 32K of SQL if all of the SQL is less than 32K. If the SQL to be logged exceeds 32K, truncate the display to the first 32K bytes.</li> </ul>
LogTable = ' <i>logTableName</i> '	<p>Optional attribute that specifies the name of the restart log table that stores checkpoint information for restarting a job.</p> <p>If running a new job, specify a new table name that is different from the name of any existing table. The Update operator then creates a new restart log table.</p> <p>If restarting a paused job, then the restart log table must exist, and the Update operator restarts the job from the last checkpoint. The restarted job will continue using the existing restart log table.</p> <p>When a job successfully completes, the operator drops the restart log table. Failure to specify a restart log table will cause the job to terminate.</p> <p>The following privileges are required on the restart log table:</p> <ul style="list-style-type: none"> <li>• SELECT</li> <li>• INSERT</li> <li>• DELETE</li> </ul> <p>The following privileges are required on the database that contains the restart log table:</p> <ul style="list-style-type: none"> <li>• DROP</li> <li>• CREATE</li> </ul> <p>The Update operator automatically maintains the restart log table. Manipulating the restart log table in any way invalidates the restart capability. If the restart log table name is not fully qualified, it is created under the user's default (logon) database. If the WorkingDatabase attribute is used, you MUST fully qualify the restart log table name, even if the restart log table is going to reside in the default (logon) database.</p>
MaxSessions = <i>maxSessions</i>	<p>Optional attribute that specifies the maximum number of sessions to log on. The MaxSessions value must be greater than 0. Specifying a value less than 1 terminates the job.</p> <p>The default is one session per available AMP. The maximum value cannot be more than the number of AMPS available.</p> <p>The main instance calculates an even distribution of the Update operator sessions among the number of instances. For example, if there are 4 instances and 16 Update operator sessions, then each instance will log on four Update operator sessions.</p>
MinSessions = <i>minSessions</i>	<p>Optional attribute that specifies the minimum number of sessions required for the Update operator job to continue.</p> <p>The MinSessions value must be greater than 0 and less than or equal to the maximum number of Update operator sessions. Specifying a value less than 1 (the default) terminates the job.</p>
NotifyExit = ' <i>inmodName</i> '	<p>Attribute that specifies the name of the user-defined notify exit routine with an entry point named _dynamn. If no value is supplied, the following default name is used:</p> <ul style="list-style-type: none"> <li>• libnotifyext.dll for Windows platforms</li> <li>• libnotifyext.dylib for the Mac platform</li> <li>• libnotifyext.so for all other UNIX platforms</li> </ul>

Syntax Element	Description
	<ul style="list-style-type: none"> <li>NOTFYEXT for z/OS platforms</li> </ul> <p>See <a href="#">Deprecated Syntax</a> for information about providing your own notify exit routine.</p>
NotifyExitIsDLL = ' <i>option</i> '	<p>Optional attribute (for z/OS systems only) that specifies whether the notify exit routine is built as a DLL (shared library) or not. Valid values are:</p> <ul style="list-style-type: none"> <li>'Yes' (or 'Y') = notify exit routine is built as a DLL (default).</li> <li>'No' or ('N') = notify exit routine is not built as a DLL.</li> </ul> <p>Specifying any other value terminates the job.</p>
NotifyLevel=' <i>notifyLevel</i> '	<p>Optional attribute that specifies the level at which certain events are reported. The valid values are:</p> <ul style="list-style-type: none"> <li>'Off' = No notification of events is provided (default)</li> <li>'Low' = 'Yes' in the Low Notification Level column</li> <li>'Med' = 'Yes' in the Medium Notification Level column</li> <li>'High' = 'Yes' in the High Notification Level column</li> </ul>
NotifyMethod = ' <i>notifyMethod</i> '	<p>Optional attribute that specifies the method to be used for reporting events. The methods are:</p> <ul style="list-style-type: none"> <li>'None' = no event logging is done (default).</li> <li>'Msg' = sends the events to a log. <ul style="list-style-type: none"> <li>On Windows, the events are sent to the event log that can be viewed using the Event Viewer. The messages are sent to the application log.</li> <li>On Solaris, AIX, and Linux platforms, the destination of the events depends on the setting specified in the /etc/syslog.conf file.</li> <li>On SLES11, the destination of the events is dependent upon the setting specified in the /etc/syslog-<i>ng</i>.conf file.</li> <li>On z/OS systems, events are sent to the job log.</li> </ul> </li> <li>'Exit' = sends the events to a user-defined notify exit routine. Row count information is in 4-byte unsigned integer values.</li> <li>'Exit64' = sends the events to a user-defined notify exit routine. Row count information is in 8-byte unsigned integer values for these events: <ul style="list-style-type: none"> <li>NMEventCheckPoint64</li> <li>NMEventPhaseIEnd64</li> <li>NMEventImportEnd64</li> <li>NMEventPhaseIIEnd64</li> <li>NMEventErrorTableI64</li> <li>NMEventErrorTableII64</li> </ul> </li> <li>'ExitEON' = sends the events to a user-defined notify exit routine. Complete Teradata object names are passed to the notify exit routine for these events: <ul style="list-style-type: none"> <li>NMEventInitializeEON</li> <li>NMEventPhaseBeginEON</li> <li>NMEventDeletesBeginEON</li> </ul> </li> </ul> <p>In addition ExitEON sends row count information in 8-byte unsigned integer values.</p>

Syntax Element	Description
NotifyString = ' <i>notifyString</i> '	<p>Optional attribute that specifies a user-defined string to precede all messages sent to the system log. This string is also sent to the user-defined notify exit routine.</p> <p>The maximum length of the string is:</p> <ul style="list-style-type: none"> <li>• 80 bytes, if the NotifyMethod is 'Exit'</li> <li>• 16 bytes, if NotifyMethod is 'Msg'</li> </ul>
Pack = <i>number</i>	<p>Optional attribute that specifies the pack factor for the job.</p> <p>Valid values are 1 through 16383. The default value is 16383.</p> <p>The pack factor applies only when the job uses the Extended MultiLoad Protocol.</p> <p>The pack factor tells the Update operator how many data records to include with the Array Insert statement. The Array Insert statement is sent to the database during the job acquisition phase.</p> <p><b>Note:</b></p> <p>When the job is loading 1 or more deferred LOB columns, the largest possible pack factor is 4096 because the database can support up to 4096 spool files per request.</p>
PauseAcq = ' <i>option</i> '	<p>Optional attribute that specifies whether to pause the Update operator job after the acquisition phase or enter the application phase. The values are:</p> <ul style="list-style-type: none"> <li>• 'Yes' (or 'Y') = Pause the Update operator job after the acquisition phase.</li> <li>• 'No' (or 'N') = Do not pause (default).</li> </ul> <p>Specifying any other value terminates the job.</p> <p>The absence of any value for the PauseAcq attribute means that the Update operator job will execute both the acquisition phase and the application phase without pausing.</p>
PrivateLogName = ' <i>logName</i> '	<p>Optional attribute that specifies the name of a log that is maintained by the Teradata PT Logger inside the public log. The private log contains all of the output provided by the Update operator.</p> <p>The private log can be viewed using the tlogview command as follows, where <i>jobId</i> is the Teradata PT job name and <i>privateLogName</i> is the value for the Update operator PrivateLogName attribute:</p> <pre>tlogview -j jobId -f privateLogName</pre> <p>By default, no diagnostic trace messages are produced. Diagnostic trace messages are produced only when the user sets a valid value for the TraceLevel attribute.</p> <p>If the private log is not specified, all of the output is stored in the public log. For more information about the tlogview command, see <a href="#">Teradata PT Utility Commands</a>.</p>
QueryBandSessInfo = ' <i>queryBandExpression</i> '	<p>Optional attribute that specifies the Query Band for the duration of the job sessions.</p> <p>The <i>queryBandExpression</i> is a set of name=value pairs, separated by a semicolon and ending with a semicolon. The user defines the Query Band expression, which will look similar to the following example:</p>

Syntax Element	Description
	<p>'org=Finance;load=daily;location=west;'</p> <p>QueryBandSessInfo may also be specified as an ARRAY attribute. For information on the rules for creating a Query Band expression, see <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i>, B035-1144 and <i>Teradata Vantage™ - SQL Data Definition Language Detailed Topics</i>, B035-1184.</p> <p>The value of the QueryBandSessInfo attribute is displayed in the Update Operator private log.</p> <p>Use of the QueryBandSessInfo attribute is subject to the following rules:</p> <ul style="list-style-type: none"> <li>• By default, Query Band is off until a valid value appears for the QueryBandSessInfo attribute.</li> <li>• If the QueryBandSessInfo attribute contains a value, the Update operator constructs the necessary SET QUERY BAND SQL and issues it as part of the Update operator SQL sessions to communicate the request to the database.</li> <li>• The Update operator does not check the Query Band expression, but passes the expression to the database as is.</li> <li>• If the version of the database against which the job is being run does not support the Query Band feature, no Query Banding will take place. However, the operator will ignore the error and run the rest of the job.</li> <li>• If there is a syntax error in the Query Band expression, the database will return an error. The Update operator will then terminate the job and report the error to the user.</li> </ul>
ReplicationOverride ='option'	<p>Optional attribute that overrides the normal replication services controls for an active session.</p> <p>Valid values:</p> <ul style="list-style-type: none"> <li>• 'On' = Override normal replication services controls for the active session.</li> <li>• 'Off' = Override of normal replication services is turned off for the active session (when change data capture is active).</li> <li>• 'None' = No override request is sent to the database (default).</li> </ul> <p>For more information, see <i>Teradata Replication Services Using Oracle GoldenGate</i> (B035-1152).</p> <p><b>Note:</b> The user ID that is logged in by the operator must have the REPLCONTROL privilege when setting the value for this attribute.</p>
RoleName = 'role name'	<p>Optional attribute that implements security in the database. The operator will prepend the value with "SET ROLE ". The syntax will be sent to the database as follows:</p> <pre>SET ROLE &lt;role name&gt;;</pre> <p>For example:</p> <pre>SET ROLE All;</pre> <p>For details of "SET ROLE" command use, see <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i>, B035-1144.</p> <p>The VARCHAR ARRAY can specify more than one value, for example:</p> <pre>VARCHAR ARRAY RoleName = [ 'role name1', 'role name2' ],</pre>

Syntax Element	Description
	<p>The operator will send the request to the database on the main control session and the auxiliary SQL session after the sessions are connected.</p> <p>The operator does not send the request on the MultiLoad protocol sessions, because the database does not allow the request to be sent on the MultiLoad protocol sessions. The operator uses the MultiLoad protocol sessions when the job uses the traditional MultiLoad protocol.</p> <p>When the job uses the extended MultiLoad protocol, the operator will send the request to the database on the data SQL sessions after the sessions are connected.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>The operator does not validate the value for this attribute. The operator passes the value to the database as is. The database will validate the value. The operator will terminate the job with an error when the validation fails.</li> <li>C-style comments are allowed in the value and will be passed to the database.</li> <li>ANSI-style comments are not supported in the value. The operator can terminate the job with a syntax error when the value contains an ANSI-style comment.</li> <li>A semicolon is not allowed in the value, because the operator allows only a single statement per request. The operator will terminate the job with an operator error when the value contains a semicolon.</li> </ul>
TargetTable = ' <i>targetTableName</i> '	<p>Required attribute that specifies the name of the Update target table to receive data from the client system.</p> <p>The table must already exist.</p>
VARCHAR TASMFATSTFAIL = ' <i>value</i> '	<p>Optional attribute that enables FASTFAIL feature.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>'Yes' or 'Y' = Enable FastFail feature. The job will terminate gracefully when it is supposed to be held by the database.</li> <li>'No' or 'N' = FastFail feature is not enabled (default). The job will appear to hang if the TASM rules dictate that a job should be held for any reason.</li> </ul>
TenacityHours = <i>hours</i>	<p>Optional attribute that specifies the number of hours that the Update operator continues trying to log on when the maximum number of load/unload operations are already running on the database.</p> <p>The default value is 4 hours. To enable the tenacity feature, the <i>hours</i> value must be greater than 0. Specifying a value of 0 disables the tenacity feature. Specifying a value of less than 0 terminates the Update job.</p>
TenacitySleep = <i>minutes</i>	<p>Optional attribute that specifies the number of minutes that the Update job pauses before retrying a logon operation when the maximum number of load/export operations are already running on the database.</p> <p>The minutes value must be greater than 0. If you specify a value less than 1, the Update operator responds with an error message and terminates the job. The default is 6 minutes.</p>

Syntax Element	Description
Tdpld = 'dbcName'	<p>Optional attribute that specifies the name of the database machine (non-mainframe platforms) or TDP (mainframe platforms) for the Update operator job.</p> <p>The <i>dbcName</i> can be up to 256 characters and can be a domain server name. If you do not specify the value for the Tdpld attribute, the operator uses the default Tdpld established for the user by the system administrator.</p>
TimeZoneSessInfo = 'timeZoneValue'	<p>Optional attribute that allows you to change the default time zone displacement for the duration of the operator's job session.</p> <p>When you provide a value for this attribute, the operator will build the SET TIME ZONE &lt;timeZoneValue&gt;; SQL request.</p> <p>The operator will send the request to the database on the main control and auxiliary SQL sessions after the sessions are connected.</p> <p>The operator does not send the request on the MultiLoad protocol sessions, because the database does not allow the request to be sent on the MultiLoad protocol sessions. The operator uses the MultiLoad protocol sessions when the job uses the traditional MultiLoad protocol.</p> <p>When the job uses the Extended MultiLoad Protocol, the operator will send the request to the database on the data SQL sessions after the sessions are connected.</p> <p>Here are some examples:</p> <ul style="list-style-type: none"> <li>• <b>Example 1:</b> This example sets the session default time zone displacement to LOCAL, which is the system default time zone:  <code>VARCHAR TimeZoneSessInfo = 'LOCAL'</code></li> <li>• <b>Example 2:</b> This example sets the session default time zone displacement to USER, which is the default time zone for the logged on user:  <code>VARCHAR TimeZoneSessInfo = 'USER'</code></li> <li>• <b>Example 3:</b> This example sets the session default time zone displacement to a simple constant time zone string expression:  <code>VARCHAR TimeZoneSessInfo = '''America Pacific'''</code></li> </ul> <p><b>Note:</b></p> <p>Any single quote character ('') inside the value must be entered as two consecutive single quote characters in a TPT job script. This ensures the correct value will be sent to the database.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>• The operator does not validate the value for this attribute. The operator passes the value to the database as is. The database will validate the value. The operator will terminate the job with an error when the validation fails.</li> <li>• C-style comments are allowed in the value and will be passed to the database.</li> <li>• ANSI-style comments are not supported in the value. The operator will terminate the job with a syntax error when the value contains an ANSI-style comment.</li> <li>• A semicolon is not allowed in the value, because the operator allows only a single statement in the "SET TIME ZONE SQL" request. The operator will terminate the job with an operator error when the value contains a semicolon.</li> </ul>

Syntax Element	Description
	<p>For more information on SET TIME ZONE SQL, see <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i>, B035-1144.</p>
TraceLevel = '/level'	<p>Optional attribute that specifies the types of diagnostic messages that are written by each instance of the operator to the public log (or private log, if one is specified using the PrivateLogName attribute). The diagnostic trace function provides more detailed information in the log file to aid in problem tracking and diagnosis.</p> <p>The trace levels are:</p> <ul style="list-style-type: none"> <li>• 'None' = TraceLevel turned off (default).</li> <li>• 'CLI' = enables the tracing function for CLI-related activities (interaction with the database)</li> <li>• 'PX' = enables the tracing function for activities related to the Teradata PT infrastructure</li> <li>• 'Oper' = enables the tracing function for operator-specific activities</li> <li>• 'Notify' = enables the tracing of activities related to the Notify feature</li> <li>• 'All' = enables tracing for all of the mentioned activities</li> </ul> <p>The VARCHAR ARRAY can specify more than one value, for example:</p> <pre>VARCHAR TraceLevel = 'CLI' VARCHAR TraceLevel = 'OPER' VARCHAR ARRAY TraceLevel = [ 'CLI' ] VARCHAR ARRAY TraceLevel = [ 'CLI', 'OPER' ]</pre> <p><b>Note:</b></p> <p>The TraceLevel attribute is provided as a diagnostic aid only. The amount and type of additional information provided by this attribute changes to meet evolving needs from release to release.</p>
TransformGroup = 'transformgroupname'	<p>Optional attribute that supports changing the active transform for Teradata Complex Data Types (CDTs). The value is the <i>&lt;udt name&gt; &lt;transform group name&gt;</i>, and the operator will prepend the hardcoded string "SET TRANSFORM GROUP FOR TYPE ". The syntax sent to RDBMS is as follows:</p> <pre>SET TRANSFORM GROUP FOR TYPE &lt;udt name&gt; &lt;transform group name&gt;;</pre> <p>For example:</p> <pre>"SET TRANSFORM GROUP FOR TYPE JSON CHARACTER SET LATIN TD_JSON_VARCHAR;"</pre> <p>The VARCHAR ARRAY can specify more than one value, for example:</p> <pre>VARCHAR ARRAY TransformGroup = [ 'JSON CHARACTER SET LATIN TD_JSON_VARCHAR',                                 'ST_Geometry TD_Geo_VARCHAR' ],</pre> <p>The operator will send the request to the database on the SQL session after the session is connected.</p> <p>The operator will send the request to the database on the main control SQL session after the session is connected.</p> <p>The operator does not send the request on the MultiLoad protocol sessions, because the database does not allow the request to be sent on the MultiLoad</p>

Syntax Element	Description
	<p>protocol sessions. The operator uses the MultiLoad protocol sessions when the job uses the traditional MultiLoad protocol. Furthermore, the operator does not send the request on the auxiliary SQL session when the job uses the traditional MultiLoad protocol, because the database does not require it. When the job uses the Extended MultiLoad Protocol, the operator will send the request to the database on the data SQL and auxiliary SQL sessions after the sessions are connected.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>The operator does not validate the value for this attribute. The operator passes the value to the database as is. The database will validate the value. The operator will terminate the job with an error when the validation fails.</li> <li>C-style comments are allowed in the value and will be passed to the database.</li> <li>ANSI-style comments are not supported in the value. The operator can terminate the job with a syntax error when the value contains an ANSI-style comment.</li> <li>A semicolon is not allowed in the value, because the operator allows only a single statement per request. The operator will terminate the job with an operator error when the value contains a semicolon.</li> </ul>
TreatDBSRestartAsFatal = ' <i>option</i> '	<p>Optional attribute that tells the operator whether or not to terminate the job when a database restart occurs.</p> <p>The TreatDBSRestartAsFatal values are:</p> <ul style="list-style-type: none"> <li>'No' ('N') = The operator will not terminate if a database restart occurs (default). The database restart will be treated as a retryable one.</li> <li>'Yes' ('Y') = The operator will terminate if a database restart occurs.</li> </ul>
UnicodePassThrough = ' <i>value</i> '	<p>Optional attribute that tells the operator to enable or disable the Unicode Pass Through feature.</p> <p>Valid values:</p> <ul style="list-style-type: none"> <li>'On' = Enable the Unicode Pass Through feature in the operator.</li> <li>'Off' = (Default) Disable the Unicode Pass Through feature in the operator.</li> </ul> <p><b>Note:</b></p> <p>When a TPT job is using the UTF8 or UTF16 session character set, the UnicodePassThrough attribute can be set to 'On' to allow the operator to load data with Unicode pass through characters.</p>
UserName = ' <i>userId</i> '	<p>Attribute that specifies the database user name.</p> <p><b>Note:</b></p> <p>Use of this attribute is not compatible with some external authentication logon methods. For more information on UserName specification requirements, see "Logon Security" in <i>Teradata® Parallel Transporter User Guide</i>, B035-2445.</p>
UserPassword = ' <i>password</i> '	Attribute that specifies the password associated with the user name.

Syntax Element	Description
	<p><b>Note:</b></p> <p>Use of this attribute is not compatible with some external authentication logon methods. For more information on password specification requirements, see “Logon Security” in <i>Teradata® Parallel Transporter User Guide</i>, B035-2445.</p>
WorkTable = 'workTable'	<p>Optional attribute that specifies the name of the work table. This must be a new table name that does not duplicate the name of any existing table, unless you are restarting a paused Update operator job.</p> <p><b>Note:</b></p> <p>If the name is not supplied, it is created by the Update operator. The name of the created table is prepended with an identifying ttname_WT, thus ensuring uniqueness (for example, ttname_WT if VARCHAR is used, or ttname1_WT, ttname2_WT, ..., ttname5_WT, if VARCHAR ARRAY is used).</p> <p><b>Note:</b></p> <p>When the job has multiple work tables and uses the Extended MultiLoad Protocol, only the first work table is created. The Update operator inserts records into the first work table during the acquisition phase.</p>
WorkingDatabase = 'databaseName'	<p>Optional attribute that specifies a database other than the logon database as the default database. The name of the database that is specified with this attribute is used in the Teradata SQL DATABASE statement that is sent by the operator immediately after connecting the two SQL sessions.</p> <p>If WorkingDatabase is not specified, the default database associated with the logged on user is assumed for all unqualified table names.</p>

## Usage Notes

### LogFile

A restart log table, which contains restart information written during job runs, is required for any execution of the Update operator. Specify a restart log table in scripts with the LogTable attribute.

Restarts (as discussed in [Staged Loading](#)) are common during staged loading operations. When additional data is available after a job is paused, the job is restarted by submitting a second script that specifies the additional data. The Update operator recognizes the job as a restart, reads the restart log to determine the status of the job, then loads the additional file.

Restarts can also occur following any unexpected error on a database. For example, if a table runs out of available space during a load operation, the job terminates, the table is paused, and a checkpoint is recorded in the restart log. Pausing the job in this way allows you to manually increase the available space, if needed, then restart the job because the load operation can restart a job from the last checkpoint in the restart log.

If you do not specify a name for *LogTable*, the Update operator automatically creates a name of the log table as follow:

```
ttname_RL
```

where *ttname* is the name of the first target table.

#### **Note:**

The value of the TargetTable attribute is truncated to the maximum number of characters for object names that the database supports, minus 3 characters before the suffix "\_RL" is appended to the target table name. This means that if the value of the TargetTable attribute is a fully qualified table name and that fully qualified name exceeds the maximum supported length of a database object, the generated name for the log table may not be what you intend. In such a case, Teradata recommends that you provide the names of the log table and not rely on the Update operator to generate the names for log table automatically.

## **TargetTable**

Required attribute that specifies the name of the Update target table to receive data from the client system. The table must already exist.

The Update operator does not support target tables defined as NoPI.

## **ErrorTable**

Update operations create two error tables for each target table. These error tables are similar to those used for the Load operator, but the Update error tables are typically named with the following suffixes to distinguish them.

Consider the following:

- Names for error tables can be defaulted or they can be explicitly named using the VARCHAR ErrorTable attribute.
- If a job generates no errors, the error tables will be empty. They are automatically dropped at the end of the job.
- If errors are generated, the tables are retained at the end of the job so error conditions can be analyzed.
- To rerun a job from the beginning, either delete the error tables, or rename them, otherwise an error message results, stating that error tables already exist.
- Conversely, if you restart a job (not from the beginning), an error tables must already exist. In other words, do not delete error tables to restart an update job.

Errors are separated into two tables, as follows:

- **Error Table (ET)** contains data errors that occur in the acquisition phase of the Update operator job.

The following types of errors are captured:

- Data conversion errors
- Data errors with the primary index fields

By default, this error table is assigned a name using the convention:

Target\_TableName\_ET

---

**Note:**

The names in the Error Table (ET) ErrorField column have a maximum supported size of 120 characters. The names can be up to 128 characters, but if a row is inserted into the Error Table (ET), The database truncates any name that exceeds 120 characters.

- **Uniqueness Violations (UV)** contains data errors that occur in the application phase of the Update operator job.

The following types of errors are captured:

- Uniqueness violations
- Field overflow on columns other than primary index fields
- Constraint errors

By default, this error table is assigned a name using the following convention:

Target\_TableName\_UV

## Error Limits

The Update operator provides the same capability as the Load operator for setting an approximate number of errors captured before a job is terminated. The number is approximate because the Update operator sends multiple rows of data at a time to the database. By the time the Update operator processes the message indicating that the error limit has been exceeded, the operator may have loaded more records into the error table than the actual number specified by the error limit.

When updating large amounts of data, it is not uncommon to encounter a data error that occurs repeatedly on each input record. Because an error can often be corrected long before errors are generated for all the records in a job run, consider using the ErrorLimit attribute to specify an approximate number of errors that can be tolerated before a job is terminated.

Note that the ET table contains errors in rows detected during the acquisition phase (the loading of data). These are commonly data conversion errors. The second table is the UV table and contains rows that are detected to be in error during the application phase of the job. These errors are commonly “uniqueness violation” errors (hence the name UV).

The ErrorLimit specification applies to each instance of the Update operator, not to all instances combined. For example, if the limit is set to 1,000, a single instance must detect that 1,000 rows were inserted into error tables to terminate the job.

This limit is specified with the *ErrorLimit* attribute. Errors are counted only during the Acquisition Phase, so the number of error rows being placed in the ET table are counted towards the number set in the *ErrorLimit* attribute. This applies to each instance of the Update operator, not to all instances combined. Therefore, if an error limit is set to 1,000, a single load instance must detect that 1,000 rows are inserted into the error tables before the job is terminated.

The error limit can also be reached at checkpoint time.

## Error Limit Examples

To illustrate how Teradata PT determines if the number of errors has reached the Error Limit, consider these examples if there are two instances running and the Error Limit has been set to 1000.

- If either instance by itself reaches 1000, it will terminate the job by returning a fatal error.
- If instance #1 processes 500 error rows and instance #2 processes 500 error rows but does not reach a checkpoint. The job will continue processing.
- If instance #1 processes 500 error rows and instance #2 processes 500 error rows but does reach a checkpoint. The total number of error rows for all instances combined is determined at checkpoint time and at the end of the Acquisition Phase. If the total of all instances exceeds the error limit at that time, the job will terminate with an error.

## Error Capture

When running Insert, Update, Delete, or Upsert requests, errors can occur due to missing or duplicate rows. When errors occur, the request is rolled back and the error is normally reported. Use the APPLY statement to specify how to handle this type of error:

- MARK means the error is to be captured and recorded.
- IGNORE means the error is not to be recorded.

Specify whether errors are marked or ignored with the following in mind:

- DUPLICATE INSERT ROWS means an attempt to insert a duplicate row.
- DUPLICATE UPDATE ROWS means an update will result in a duplicate row.
- MISSING DELETE ROWS means an attempt to delete a row that is missing.
- MISSING UPDATE ROWS means an attempt to update a row that is missing.

## WorkTable

Update operations create one work table for each target table, using the following naming convention:

Target\_TableName\_WT

As with the error tables, these default names can be overridden. Use the VARCHAR WorkTable attribute.

## Auto-Generation of Error and Work Tables

Some tables are created during the execution of a job, and others must be created by the user before the job begins. The log table is created automatically when you run the Update job script. Target tables must exist on the database when the Update operator job is executed.

If you have not specified error and work tables (specifying them is optional), the Update operator automatically creates the names of the error and work tables.

If you specify only one target table, use VARCHAR syntax, as follows:

```
VARCHAR TargetTable = 'ttname'
```

If you specify more than one target table, you can also use VARCHAR syntax, as follows:

```
VARCHAR TargetTable = ['table1', 'table2', ..., 'tableN']
```

You can, if you wish, use ARRAY syntax, as follows:

```
VARCHAR ARRAY TargetTable = ['table1', 'table2', ..., 'tableN']
```

---

**Note:**

Using the ARRAY keyword in assigning an array value is optional.

---

In either case, the Update operator automatically creates the names of the error tables and the work table, associated with a target table, as follows:

- The first error table is ttname\_ET
- The second error table is ttname\_UV
- The work table is ttname\_WT

where ttname is the name of the corresponding target table.

---

**Note:**

The value of the TargetTable attribute is truncated to the maximum number of characters for object names that the database supports, minus 3 characters before the suffixes "\_ET" and "\_UV" are appended to target table names. This means that if the value of the TargetTable attribute is a fully qualified table name and that fully qualified name exceeds the maximum supported length of a database object, the generated names for the error tables may not be what you intend. In such a situation, Teradata recommends that you provide the names of the error and work tables and not rely on the Update operator automatically generating the names for these tables.

You cannot specify more error or work tables than there are target tables already defined, but you may specify fewer.

If the following is specified when no other error or work table specifications exist,

```
VARCHAR TargetTable = ['targtable1', 'targtable2', 'thirdtable']
```

the Update operator creates the following error tables and work tables:

```
targtable1_ET
targtable1_UV
targtable1_WT
targtable2_ET
targtable2_UV
targtable2_WT
thirdtable_ET
thirdtable_UV
thirdtable_WT
```

Notice that each set of two error tables and one work table belong to a particular target table; the naming convention preserves the uniqueness of the tables associated with a target table.

If you specify the following, the Update operator creates the necessary missing table names:

```
VARCHAR TargetTable = ['ttname1', 'ttname2', 'ttname3']
VARCHAR ErrorTable1 = ['error_1']
VARCHAR ErrorTable2 = ['error_2']
VARCHAR WorkTable = ['work_1', 'work_2']
```

If you specify more error table names or work table names than there are target table names, the Update operator issues an error message and terminates the job.

## Extended MultiLoad Protocol

The Extended MultiLoad Protocol is an alternative way to load data into a target table when the traditional MultiLoad protocol cannot be used because of a restriction on the target table.

The database decides if a job uses the traditional or Extended MultiLoad Protocol. If the job can use the traditional MultiLoad protocol, the job uses it. If the job cannot use the traditional MultiLoad protocol, the job uses the Extended MultiLoad Protocol. The user does not select which protocol the job uses.

Below are the jobs that use the Extended MultiLoad Protocol:

- The target table has a join index, hash index, unique secondary index, or referential integrity.
- The target table has no primary index.

- The target table has a Large Object (LOB) columns. A LOB column can be a Binary Large Object (BLOB) or a Character Large Object (CLOB).
- The target table has XML or JSON columns.

The following table shows differences between the traditional and Extended MultiLoad Protocols from the Update operator's perspective:

For...	Traditional MultiLoad Protocol . . .	Extended MultiLoad Protocol . . .
Sessions	uses MultiLoad sessions.	uses SQL sessions.
DML statements	supports multiple DML statement per DML group.	supports only DML statement or an Upsert statement per DML group.
Buffer size (in KB)	supports up to 16384 KB.	N/A
Pack factor	N/A	supports up to 16383 data records per an Array Insert statement.
Work table	can use 1 or more work tables.	uses only the first work table.
Delete task	supports delete task.	does not support delete task.
Acquisition phase	uses data parcels to populate the staging work table.	uses the Array Insert statement to populate the staging work table.
Application phase	uses the EXEC MLOAD request to apply the rows in the staging work table to the target table.	uses the Merge Into request to apply the rows in the staging work table to the target table.
Performance	is faster.	is slower.

The following are restrictions on the Extended MultiLoad Protocol. The target table cannot be a:

- Column partitioned table
- Replication group
- Table with an identity column defined as a primary index
- Table with a trigger

## Normalized Tables

The Update operator does not support normalized tables when the job uses the traditional MultiLoad protocol. The Update operator does support normalized tables when the job uses the Extended MultiLoad Protocol.

A normalized table is a table that has been created using the NORMALIZE clause in the CREATE TABLE statement. To normalize means to combine two period values that meet or overlap.

## Restrictions and Limitations

This section describes the restrictions and limitations when using the Update operator.

## Table Characteristics and Considerations

When using the traditional MultiLoad protocol, the target table cannot have any of the following characteristics:

- Join, hash, or secondary indexes
- No primary index
- Foreign key references
- Referential integrity
- Triggers
- Cannot be a normalized table
- Cannot be a column partitioned table
- Cannot be a temporal table
- Cannot be in a replication group

The Extended MultiLoad Protocol can be used to load target tables that cannot be loaded using the traditional MultiLoad protocol. The exceptions are:

- Cannot be a column partitioned table
- Cannot be a temporal table
- Cannot be in a replication group
- Cannot have an identity column defined as a primary index

## Data Types

- The Update operator can load CLOB/BLOB/JSON/XML in inline and deferred modes on all non-z/OS platforms.
  - On z/OS, the operator can load CLOB/BLOB/JSON/XML in inline mode if the size is less than or equal to 64,000 bytes. It cannot load CLOB/BLOB/JSON/XML in deferred mode.
- TPT does not support the ST\_Geometry data type in the TPT schema. But, the Update operator *can* load data to ST\_Geometry columns if the job specifies CLOB or BLOB in the TPT schema.
- UDT data can be loaded in its external type format.
- Error and Work Tables

The table names for the ErrorTable1, ErrorTable2, and WorkTable attributes must be new tables unless you are restarting a paused Update operator job.

## Enhanced LOB Support

The Update operator supports enhanced LOB functionality in TPT script mode.

The operator does not support enhanced LOB functionality in TPT API mode.

When the TPT job has one or more LOBs (inline or deferred) and the schema size is greater than 1 MB, the Update operator will use 1 MB message size for sending data to the database. The database has a limit of 1 MB message size for such jobs in the Extended MultiLoad protocol.

For more information, see [Enhanced LOB Support](#).

## Other Notes

- One Update operator job can load up to 5 database tables for an import task. A single table is supported for a delete task.
- By default, duplicate rows are not discarded by the database for SET tables. Duplicate rows are inserted into the application phase error table. Use the DML option, "IGNORE DUPLICATE ROWS," to override the default if you do not want the duplicate rows to be inserted into the application phase error table.
- Duplicate rows are not discarded by the database for MULTISET tables; they are inserted into the target tables. The DML options, "MARK DUPLICATE ROWS" and "IGNORE DUPLICATE ROWS" cannot override this behavior. Duplicate rows are allowed for MULTISET tables.
- Only the Teradata SQL INSERT, UPDATE, DELETE, and UPSERT statements are supported; any other Teradata SQL statements are not supported.
- Data can be loaded into empty or populated database tables.
- A delete task cannot be used on a view.
- The target tables are locked until the application phase is complete.
- The target tables cannot have column names that are the same as the error table column names, because the application phase error table includes a mirror image of the target table, preceded by the error information. For more information on the application error table, see [ErrorTable](#).
- You can pause the job for an import task. You cannot pause the job for a delete task.
- The target tables cannot be rolled back after the application phase is complete.
- The Update operator requires one database load slot when the job uses the traditional MultiLoad protocol. When the job uses the Extended MultiLoad Protocol, the Update operator requires one database load slot if the configurable database control field, MLOADXUtilityLimits, is set to FALSE. If MLOADXUtilityLimits is set to TRUE, the Update operator requires one database Extended MultiLoad Protocol slot when the job uses the Extended MultiLoad Protocol.

## Job Options

### Checkpointing and Restarting

The Update operator takes a checkpoint at the beginning and the end of the acquisition phase. More granular checkpoints during the acquisition phase can be specified using the -z option of the **tbuild** command, which specifies checkpoint intervals in terms of seconds. For example:

```
tbuild -f <file name> -z 120
```

In this example, the **-f** option specifies the name of the script that is input to **tbuild**, and the **-z** option indicates that a checkpoint is taken every 120 seconds.

For more information about checkpoints and restarts, see the *Teradata Parallel Transporter User Guide* (B035-2445).

The Update operator cannot be rolled back. Once changes are applied to target tables in the application phase, a job can only move forward.

Since a target table cannot be returned to its original state, it is advisable to archive tables prior to running Update operations against them.

To restart a job from the beginning (and bypass the restart log table) do the following:

1. Drop the restart log table.
2. Drop the error and work tables.
3. Drop the checkpoint files.

To discontinue an Update operation and drop the target tables, do the following:

1. Drop the restart log table.
2. Drop the error and work tables.
3. Drop the target tables.

## Upserts

Upsert operations update rows if they already exist in a table, and insert new rows they do not already exist. Specified an Upsert operation in an APPLY statement using the expression:

```
INSERT FOR MISSING UPDATE ROWS;
```

An Upsert operation fails only if both the update and the insert fail.

Also, the following specification is usually included for Upsert operations in anticipation of missing rows:

```
IGNORE MISSING UPDATE ROWS
```

## AMP Usage

### Down AMPs

The impact of down AMPs on the Update operator depends on the following:

- The number of AMPs that are down, either logically or physically, in a cluster
- The operational phase of the Update tasks when the down AMP condition occurs

- Whether the target tables are fallback or nonfallback

The following table describes the impact of down AMPs on Update operator tasks on fallback and nonfallback tables.

AMP Usage		
Condition 1	Condition 2	Result
All of the target tables are fallback	Not more than one AMP is down	<p>Update operator tasks continues to run as long as there is not more than one AMP down, either logically or physically, in a cluster.</p> <p>The down AMP does not participate in the application phase if:</p> <ul style="list-style-type: none"> <li>• The AMP goes down before the Update operator tasks enter the application phase, and the AmpCheck attribute is set to 'None'</li> <li>• Certain I/O errors occur during the application phase</li> </ul>
	Two or more AMPs are down	<p>Update operator tasks do not run, or terminate if two or more AMPs are down, either logically or physically, in a cluster.</p> <p><b>Note:</b> In the application phase, if AMPs are down to the extent that data on the DSU is corrupted, then you must restore the affected tables.</p>
One or more of the target tables is nonfallback	One or more AMPs are down	<p>Update operator tasks terminate and you cannot restart until all of the AMPs are back up.</p> <p><b>Note:</b> The Update operator also terminates if I/O errors corrupt the target tables in the application phase.</p>

## Nonparticipant AMPs

An AMP can become nonparticipant for an Update operator task one of the following ways:

- When any AMP is down at the end of the acquisition phase/beginning of the application phase, the associated AMP becomes a nonparticipant if the AmpCheck 'None' option is specified. Because the Update operator does not run after the acquisition phase if an AMP is down and the target table is nonfallback, an AMP can become a nonparticipant only if the target table is defined as having fallback protection. The AmpCheck 'Apply' and 'All' options prevent the occurrence of nonparticipant AMPs in this situation.
- When I/O errors occur in certain Update operator tables during the application phase, the associated AMP becomes a nonparticipant if the I/O recovery operation stops the Update operator task.
- When a head/disk assembly (HDA) fails during the application phase, the associated AMP becomes a nonparticipant but it returns after the disk is replaced and the Disk Copy and Table Rebuild utilities are run.

In effect, the Update operator treats a nonparticipant AMP as if it were a down AMP. Thus, the Update operator does not run if a cluster has any combination of more than one AMP that is:

- Down
- Offline
- Nonparticipant

And, if more than one AMP in a cluster becomes a nonparticipant during the application phase, the Update operator tasks cannot continue. The target tables are considered unusable, and must be recovered from archives.

## One-AMP Systems

The Update operator cannot run on a one-AMP database system, or on a multi-AMP system configured with one-AMP clusters. Any attempt to run Update in this environment triggers the database to immediately reject the request, abort the job, and issue a diagnostic message stating that a one-AMP system is not supported.

## VARCHAR ARRAY Tables

Target, error, and work table names can be specified in terms of VARCHAR ARRAY types if specifying more than one table, using the following syntax:

```
VARCHAR TargetTable = ['table1', 'table2', ..., 'tableN']
```

---

### Note:

Using the ARRAY keyword in assigning an array value is optional.

---

You cannot specify more error or work tables than there are target tables defined, but you may specify fewer. If fewer error/work table names are defined than target tables, the Update operator creates a name for the error/work table:

- The first error table is *ttnamEN\_ET*
- The second error table is *ttnamEN\_UV*
- The work table is *ttnamEN\_WT*, where *ttnamEN* is the name of the corresponding target table

---

### Note:

Target table names are truncated to the maximum number of characters for table names on the database minus 3 characters before the suffixes "\_ET", "\_UV", or "\_WT" are appended to target table names.

---

For example, if the following is specified when no other error/work table specifications exist,

```
VARCHAR TargetTable = ['targtable1', 'targtable2', 'thirdtable']
```

the Update operator creates the following error tables and work tables:

```
targtable1_ET
targtable1_UV
targtable1_WT
targtable2_ET
targtable2_UV
targtable2_WT
thirdtable_ET
thirdtable_UV
thirdtable_WT
```

Note that each set of two error tables and one work table belong to a particular target table; the naming convention preserves the uniqueness of the associated target table.

If you specify the following, the Update operator creates the necessary missing table names:

```
VARCHAR TargetTable = ['ttname1','ttname2','ttname3']
VARCHAR ErrorTable1 = ['error_1']
VARCHAR ErrorTable2 = ['error_2']
VARCHAR WorkTable = ['work_1','work_2']
```

If you specify more error table names or work table names than there are target table names, the Update operator issues an error message and terminates the job.

## Dropped Tables

Some tables are created during the execution of a job, and others must be created by the user before the job begins. With the Update operator, target tables must exist on the database when the Update operator job is executed. The log table is created automatically when you run the Update job script. Error tables and work tables are created by the database.

Error tables are dropped by the Update operator during the cleanup phase if no errors are detected during the acquisition phase or the application phase. The work table is dropped by the Update operator during the cleanup phase if no errors are detected during the acquisition phase or the application phase. The log table is dropped by the Update operator after the job completes successfully.

If a job terminates abnormally, then the log, error, and work tables may not be dropped, depending on where in the job the termination occurs. If you want to restart the job from scratch, you must manually drop these tables by running a BTEQ script.

Care must be taken if the target tables are manually dropped using a BTEQ script. If something goes wrong with an Update operator job, and you drop the target table manually and then try to rerun the job,

you may lose the original data. This is because all rows are actually placed into worktables and they remain there until the Update operator job reaches the application phase, at which time the rows are copied to the real target tables.

## Operational Considerations

### Space Requirements

Always estimate the final size of the Update target table, and ensure that the destination database has enough space to accommodate the Update job.

If the database that owns the Update target table, log table, error tables, or work table runs out of space, the database returns an error message and the Update operator pauses the Update operator job. When this happens, you must allocate more space to the database before you can restart the job.

### Sessions and Instances

The Update operator will connect one main (control) SQL session and one auxiliary SQL session, in addition to the data sessions. The main SQL session is responsible for executing SQL statements pertaining to utility work. The auxiliary session is used for creating and maintaining a restart log table for recovery purposes.

A minimum and a maximum number of sessions (the session limits) can be specified for the Update operator.

Consider the following usage notes:

- The maximum sessions connected can never exceed the number of available AMPs in the system, even if a larger number is specified.
- The default is one session per available AMP.
- For the MinSessions attribute, the minimum specification is one.
- The MaxSessions attribute can be set to a number smaller than the number of AMPs on the database server if fewer sessions are suitable for the job.
- Network protocol software might also impose limits on workstation-attached systems.
- Platform limits for maximum sessions per application differ:
  - On a mainframe-attached z/OS client system, use the TDP SET MAXSESSIONS command to specify a platform limit.
  - On workstation-attached client systems for UNIX, Linux, and Windows systems, this value is defined in the CLI file, clispb.dat, under the *max\_num\_sess* variable.
  - On mainframe-attached z/OS client systems, this value is defined in the HSHSPB parameter under the IBCSMAX setting.

The *max\_num\_sess* value in the clispb.dat file (or HSHSPB) specifies the total number of sessions allowed to be connected by a single application at one time. The *max\_num\_sess* pertains to all sessions connected, both SQL and data loading.

## Limits on Update Jobs

The Update operator requires one database load slot when the job uses the traditional MultiLoad protocol.

The number of concurrent load slots is, however, configurable in the database environment using the same MaxLoadTasks field control used by FastLoad, FastExport, and MultiLoad. For example, one Teradata PT Update job equates to one MultiLoad .IMPORT step in a MultiLoad job in the count for MaxLoadTasks.

When the job uses the Extended MultiLoad Protocol, the Update operator requires one database load slot if the configurable database control field, MLOADXUtilityLimits, is set to FALSE.

If MLOADXUtilityLimits is set to TRUE, the Update operator requires one database Extended MultiLoad Protocol slot when the job uses the Extended MultiLoad Protocol.

The number of concurrent Extended MultiLoad Protocol slots is, however, configurable in the database environment using the MaxMLOADXTasks field control. The number of concurrent Extended MultiLoad Protocol slots counts when the MLOADXUtilityLimits field control is set to TRUE.

Sample scenarios include:

- Scenario #1: Assume the MaxLoadTasks field control is set to 15 and the MLOADXUtilityLimits field control is set to FALSE. The database will allow a maximum of 15 concurrent jobs that use the FastLoad, FastExport, traditional MultiLoad, and Extended MultiLoad Protocols.

---

### Note:

The value for the MaxMLOADXTasks field control is ignored, because the MLOADXUtilityLimits field control is set to FALSE.

- 
- Scenario #2: Assume the MaxLoadTasks field control is set to 15, the MLOADXUtilityLimits field control is set to TRUE, and the MaxMLOADXTasks field control is set to 60. The database will allow a maximum of 15 concurrent jobs that use the FastLoad, FastExport, and traditional MultiLoad protocols, and the database will allow a maximum of 60 concurrent jobs that use the Extended MultiLoad Protocol.

For more information, see “DBS Control Utilities” in *Teradata Vantage™ - Database Utilities*, B035-1102.

## Temporal Tables

The Update operator can load data into temporal tables using the Extended MultiLoad Protocol. The job automatically uses the Extended MultiLoad Protocol when the job is loading into temporal tables. The temporal tables can be of two kinds: 1) Teradata Temporal or 2) ANSI Temporal.

The column schema USINGEXTENSION option must be used to load temporal tables. The Update operator does not validate the value for the USINGEXTENSION option. The database validates the value.

## Example 1 – Teradata Temporal Syntax

Here is a Teradata Temporal example of the target temporal table definition with a VALIDTIME column named JOBDURATION:

```
CREATE MULTISET TABLE TARGET_EMP_TABLE(
    EMP_ID      INTEGER,
    EMP_NAME    CHAR(30),
    EMP_DEPT    INTEGER,
    JOBDURATION PERIOD(DATE) NOT NULL AS VALIDTIME)
PRIMARY INDEX(EMP_ID);
```

Here is an example of the Teradata PT schema definition for the sample target Teradata temporal table definition:

```
DEFINE SCHEMA EMPLOYEE_SCHEMA
DESCRIPTION 'SAMPLE EMPLOYEE SCHEMA'
(
    EMP_ID      INTEGER,
    EMP_NAME    CHAR(30),
    EMP_DEPT    INTEGER,
    JOBDURATION PERIOD(DATE) USINGEXTENSION('AS VALIDTIME')
);
```

In the sample Teradata PT schema definition, the USINGEXTENSION option is specified for the JOBDURATION column. The USINGEXTENSION option has a value of 'AS VALIDTIME'.

The job must specify the DML statements with a temporal qualifier (SEQUENCED, NONSEQUENCED, or CURRENT). Here is a sample DML statement with the SEQUENCED temporal qualifier:

```
SEQUENCED VALIDTIME INSERT INTO
TARGET_EMP_TABLE (:EMP_ID, :EMP_NAME, :EMP_DEPT, :JOBDURATION);
```

The Update operator will build the USING clause and send the USING clause to the database. Here is a sample USING clause:

```
USING EMP_ID(INTEGER), EMP_NAME(CHAR(30)), EMP_DEPT(INTEGER),
      JOBDURATION(PERIOD(DATE) AS VALIDTIME)
SEQUENCED VALIDTIME INSERT INTO TARGET_EMP_TABLE
  (:EMP_ID,          :EMP_NAME,          :EMP_DEPT,  :JOBDURATION);
```

The database needs the USINGEXTENSION value to create the temporal NoPI staging table. During the Acquisition phase, data are populated into the staging table. After the Acquisition phase, the data are

merged from the staging table to the temporal target table in the Application phase. After the Application phase, the staging table can be dropped.

## Example 2 – ANSI Temporal Syntax

Here is a ANSI Temporal example of the target temporal table definition with a derived period column JOBDURATION as a VALIDTIME column:

```
CREATE MULTISET TABLE TARGET_EMP_TABLE(
    EMP_ID      INTEGER,
    EMP_NAME    CHAR(30),
    EMP_DEPT    INTEGER,
    JOB_START   DATE NOT NULL,
    JOB_END     DATE NOT NULL,
    PERIOD FOR JOBDURATION(JOB_START, JOB_END) AS VALIDTIME)
PRIMARY INDEX(EMP_ID);
```

Here is an example of the Teradata PT schema definition for the sample target ANSI temporal table definition:

```
DEFINE SCHEMA EMPLOYEE_SCHEMA
DESCRIPTION 'SAMPLE EMPLOYEE SCHEMA'
(
    EMP_ID      INTEGER,
    EMP_NAME    CHAR(30),
    EMP_DEPT    INTEGER,
    JOB_START   INTDATE,
    JOB_END     INTDATE USINGEXTENSION('PERIOD FOR JOBDURATION
                                (JOB_START, JOB_END)
AS VALIDTIME')
);
```

In the sample Teradata PT schema definition, the USINGEXTENSION option is specified for the JOB\_END column. The USINGEXTENSION option has a value of 'PERIOD FOR JOBDURATION (JOB\_START, JOB\_END) AS VALIDTIME'.

The job must specify the DML statements with a temporal qualifier (SEQUENCED, NONSEQUENCED, or CURRENT). Here is a sample DML statement with the SEQUENCED temporal qualifier:

```
SEQUENCED VALIDTIME INSERT INTO
TARGET_EMP_TABLE (:EMP_ID, :EMP_NAME, :EMP_DEPT, :JOB_START, :JOB_END);
```

The Update operator will build the USING clause and send the USING clause to the database. Here is a sample USING clause:

```
USING EMP_ID(INTEGER),EMP_NAME(CHAR(30)), EMP_DEPT(INTEGER),
      JOB_START(DATE), JOB_END(DATE PERIOD FOR JOBDURATION(JOB_START,
      JOB_END) AS VALIDTIME)
SEQUENCED VALIDTIME INSERT INTO TARGET_EMP_TABLE (:EMP_ID,
      :EMP_NAME, :EMP_DEPT, :JOB_START, :JOB_END);
```

The database needs the USINGEXTENSION value to create the temporal NoPI staging table. During the Acquisition phase, data are populated into the staging table. After the Acquisition phase, the data are merged from the staging table to the temporal target table in the Application phase. After the Application phase, the staging table can be dropped.

## Notes

- The Update operator cannot load data into temporal tables using the traditional MultiLoad protocol, because the database does not support it.
- If the USINGEXTENSION option is not specified, the job will continue and the job will be a non-temporal loading.
- The database allows only one VALIDTIME column when loading temporal tables in the Extended MultiLoad Protocol. The job will terminate with an error when there are multiple columns in the Teradata PT schema with the USINGEXTENSION option.
- The value for the USINGEXTENSION option is case-insensitive.
- If an USINGEXTENSION value is specified for non-temporal columns (like INTEGER, CHAR, etc), the Update operator will send the value to the database when the job is using the Extended MultiLoad Protocol. The database will validate the value and return an error if the value is invalid.
- If an USINGEXTENSION value is specified for non-temporal columns, the Update operator will ignore the value and will not send value to the database when the job is using the traditional MultiLoad protocol.

## Delete Task Option

The Delete Task option is unique to the Update operator. It deletes rows more quickly than using a single DELETE SQL statement.

When the DeleteTask attribute is set to 'Y', rows are deleted from a single table based on values other than a unique primary index (UPI) equality condition. A Delete Task option uses a single session and a single instance.

The Delete Task option is a good choice when a large percentage of rows in a table need to be deleted, such as deleting all rows with a transaction date prior to a specified date.

The Delete Task option operates very similarly to the standard DELETE statement in the Update operator, with the following differences:

- Deleting based on non-index columns is normal for the Delete Task option.
- Deleting based on a primary index, although possible, has certain limitations:
  - An equality test of a UPI value is not permitted.
  - An inequality test of a UPI value is permitted.
  - An equality test of a NUPI value is permitted.
- A single DELETE statement is used in the APPLY statement.
- The Delete Task option does not include an acquisition phase because there are no varying input records to apply.
- The application phase reads each target block and deletes qualifying rows.
- Altered blocks are written back to disk.

When the Delete Task option is specified, the Update operator functions as a standalone operator, that is, not as the usual consumer operator that reads from a source data stream. The exception is when the Delete Task is invoked by an APPLY statement that includes a WHERE clause, and the source data stream contains only a single row. In this case, the Update operator with the Delete Task option still functions as a consumer operator.

The following rules apply to a Delete Task operation regardless of whether it functions as a standalone or consumer operator:

- Only one session is connected.
- Only one instance is specified.
- Only one DML group is specified.
- Only one DML statement in the DML group is specified.
- Only a single DELETE statement is used.
- Only one target table is specified.
- The first error table is not used and is ignored.
- Only one data record is provided if using a WHERE clause.

## Using Delete Task

Using the Delete Task option of the Update operator can run as either a consumer operator or as a standalone operator, depending on the construction of the APPLY statement:

- As a standalone operator that does *not* attempt to retrieve a row from the data stream:

```
APPLY
    <SQL DELETE statement>
```

- As a consumer operator that attempts to read a row from the data stream:

```
APPLY
  <SQL DELETE statement>
  SELECT FROM ...
```

Specify a single DELETE statement in the APPLY statement:

```
APPLY
(
  'DELETE FROM TABLE xyz;'
) ;
```

In this case, the Update operator runs as a standalone operator. (The Update operator is a consumer operator and there is no producer operator, so there is no SELECT statement).

You can also specify a DELETE statement in which information in the DELETE requires some data. In this case, the APPLY needs a SELECT statement:

```
APPLY
(
  'DELETE FROM TABLE xyz WHERE Field1 = :Field1;'
)
SELECT * FROM OPERATOR (FILE_READER [1] . . .
; ;
```

In this case, the Update operator is running as a consumer operator and it requires exactly one row of data. That row of data is passed to the database, which extracts the data from the column as specified in the WHERE clause.

Another example, similar to the first one, is where the Update operator runs as a standalone operator but the DELETE statement has a WHERE clause:

```
APPLY
(
  'DELETE FROM TABLE xyz WHERE Field1 = ''abc'';'
) ;
```

In this case, there is a WHERE clause, but the information in the WHERE clause does not require data from a producer operator.

---

**Note:**

When the Update operator runs as a standalone operator, no schema is necessary. That is, you do not need to define a schema using the DEFINE SCHEMA statement. This is because no data is needed from a producer operator for the job.

---

## Why Choose the Delete Task Option?

A simple SQL DELETE statement can usually accomplish the same result as the Delete Task option, but the Delete Task option is usually preferred because it requires fewer system resources, and therefore generally performs better than an SQL DELETE.

- The Delete Task option does not use the Transient Journal so it uses less disk space, requires less I/O, and runs faster.
- The Delete Task option aborts without rollback as opposed to the SQL DELETE, which uses the Transient Journal to roll back all changes. The Delete Task option only moves forward.

## Example 1: Delete Task Option

Following is an example of a job that uses the Delete Task option to delete rows from a table named Customer, where customer\_number is less than the hard-coded job script value of 100000:

```

DEFINE JOB DELETE_TASK
DESCRIPTION 'Hard-coded DELETE FROM CUSTOMER TABLE'
(
  DEFINE OPERATOR UPDATE_OPERATOR
  DESCRIPTION 'Teradata PT UPDATE OPERATOR'
  TYPE UPDATE
  SCHEMA *
  ATTRIBUTES
  (
    VARCHAR TargetTable  = 'Customer',
    VARCHAR TdpId       = @Tdpid,
    VARCHAR UserName     = @Userid,
    VARCHAR UserPassword = @Pwd,
    VARCHAR AccountId,
    VARCHAR LogTable     = 'DeleteTask_log',
    VARCHAR DeleteTask   = 'Y'
  );
  APPLY
  (
    'DELETE FROM CUSTOMER WHERE CUSTOMER_NUMBER LT 100000;'
  )
  TO OPERATOR( UPDATE_OPERATOR [1] );
);

```

Notice the following about this script:

- Setting the attribute DeleteTask to 'Y' makes this execution of the Update operator a Delete Task.
- The example uses the hard-coded value of *100000* in the deletion criterion.

For increased flexibility, this value *could* be specified as a job variable, such as:

```
'DELETE FROM CUSTOMER WHERE CUSTOMER_NUMBER LT ' || @Custno || ';
```

The value for the variable in this expression can come from a job variable file, or it can come from the command line:

```
tbuild -f <filename> -u "Custno = '100000'"
```

- The script still requires the SCHEMA \* clause even though the Update operator is functioning as a standalone operator.
- The APPLY statement must specify a single SQL DELETE statement.
- The LogTable attribute is always required for the Update operator.

For additional information about using variables, see the *Teradata Parallel Transporter User Guide* (B035-2445).

## Example 2: Delete Task Option

The following example accomplishes the same purpose as the previous Delete Task example, but with a slightly different technique. Rather than hard-coding values in the deletion criterion, the value in this example is supplied from an external file by the DataConnector operator and a data stream. The SQL host variable (:CustNo) represents the value in the DELETE statement.

In this case, the Update operator, used as a Delete Task, is a consumer operator because it receives input from a data stream. Differences between this approach and the first example are shown in bold text.

```
DEFINE JOB DELETE_TASK_PARAM
DESCRIPTION 'External File DELETE FROM CUSTOMER TABLE'
(
  DEFINE SCHEMA CUST_NUM_SCHEMA
  DESCRIPTION 'CUSTOMER NUMBER SCHEMA'
  (
    Cust_Num INTEGER
  );
);

DEFINE OPERATOR UPDATE_OPERATOR
DESCRIPTION 'Teradata PT UPDATE OPERATOR'
TYPE UPDATE
SCHEMA CUST_NUM_SCHEMA
ATTRIBUTES
(
```

```

VARCHAR TargetTable = 'Customer',
VARCHAR TdpId = @Tdpid,
VARCHAR UserName = @Userid,
VARCHAR UserPassword = @Pwd,
VARCHAR AccountId,
VARCHAR LogTable = 'DeleteTask_log',
VARCHAR DeleteTask = 'Y'
);

DEFINE OPERATOR DATA_PRODUCER
DESCRIPTION 'DATA CONNECTOR OPERATOR'
TYPE DATACONNECTOR PRODUCER
SCHEMA CUST_NUM_SCHEMA
ATTRIBUTES
(
  VARCHAR OpenMode = 'Read',
  VARCHAR Format = 'Formatted',
  VARCHAR IndicatorMode,
  VARCHAR FileName = 'Single_Row_File'
);
APPLY
(
  'DELETE FROM CUSTOMER WHERE CUSTOMER_NUMBER LT :CustNo')
TO OPERATOR (UPDATE_OPERATOR [1])
SELECT * FROM OPERATOR (DATA_PRODUCER[1]);
);

```

## Explanation

Notice the following in this script:

- When using this approach, the DataConnector operator write a single row to the data stream so its input file contains a single record.
- A schema must be defined in order for the input value to be read.
- An SQL host variable (:CustNo) is used in the DELETE statement.
- The colon (:) symbol prefixed to CustNo specifies to the database that the value comes from a source external to the SQL DELETE statement.

# Notify Exit Routines

## Notify Exit Routines

The Load, Export, Update, and Stream operators support notify exit routines. A notify exit routine specifies a predefined action to be performed whenever certain significant events occur during a job, for example, whether a load job succeeds or fails, how many records are loaded, what the return code is for the failed job, and so on. Only the main instance sends a notify event.

Jobs accumulate operational information about specific events that occur. If the NotifyMethod attribute specifies the Exit method, when the specific events occur the operator calls the named notify exit routine and passes the following to it:

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

**Note:**

The existing TPT function names should not be used in User's Notify Exits. Using the existing TPT function names in User's exits might cause the normal operation of the TPT job to behave with unexpected results and prevent requests from completing normally.

## Export Operator Events

The Export operator supports notify exit routines. A notify exit routine specifies a predefined action to be performed whenever certain significant events occur during a job, for example, whether a job succeeds or fails, how many records are exported, what the return code is for the failed job, and so on. Only the main instance sends a notify event.

The Export operator job accumulates operational information about specific events that occur during a job. If the Export operator job script includes the NotifyMethod attribute with the Exit method specification, then when the specific events occur, the Export operator calls the named notify exit routine and passes to it:

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

The following table lists the event codes and describes the data that the Export operator passes to the notify exit routine for each event. The information in this table is also sent to the system log.

**Note:**

Ensure that notify exit routines *ignore* invalid or undefined event codes, and that they *do not* cause the operator to terminate abnormally.

Event	Event Code	Event Description and Data Passed to the Notify Exit Routine
Initialize	0	<p>Signifies successful processing of the notify feature:</p> <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>• Operator ID—4-byte unsigned integer</li> <li>• Operator name length—4-byte unsigned integer</li> <li>• Operator 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> <li>• Operator handle—4-byte unsigned integer</li> </ul>
Database Restart	9	<p>Signifies that the Export operator received a crash message from the database or from the CLIV2:</p> <ul style="list-style-type: none"> <li>• No data accompanies the restart event code</li> </ul>
CLIV2 Error	10	<p>Signifies that the Export operator received a CLIV2 error:</p> <ul style="list-style-type: none"> <li>• Error code—4-byte unsigned integer</li> </ul>
Database Error	11	<p>Signifies that the Export operator received an error that will produce an exit code of 12:</p> <ul style="list-style-type: none"> <li>• Error code—4-byte unsigned integer</li> </ul> <p><b>Note:</b> Not all errors cause this event. A 3807 error, for example, while trying to drop or create a table does not terminate the Export operator.</p>
Exit	12	<p>Signifies that the Export operator is terminating:</p> <ul style="list-style-type: none"> <li>• Exit code—4-byte unsigned integer</li> </ul>
Export Begin	31	<p>Signifies that the Export operator is about to begin the export task by opening the export file:</p> <ul style="list-style-type: none"> <li>• No data accompanies the export begin event code</li> </ul>
Request Submit Begin	32	<p>Signifies that the Export operator is about to submit the SELECT request to the database:</p> <ul style="list-style-type: none"> <li>• Request length—4-byte unsigned integer</li> <li>• Request text—32,000-character (maximum) array</li> </ul>
Request Submit End	33	<p>Signifies that the Export operator has received the response to the SELECT request:</p> <ul style="list-style-type: none"> <li>• Statement count—4-byte unsigned integer</li> <li>• Block count—4-byte unsigned integer</li> </ul>
Request Fetch Begin	34	<p>Signifies that the Export operator is about to fetch the results of the SELECT request</p> <ul style="list-style-type: none"> <li>• No data accompanies the request fetch begin event code</li> </ul>

Event	Event Code	Event Description and Data Passed to the Notify Exit Routine
File or OUTMOD Open	35	<p>Signifies that the Export operator is about to open an output or OUTMOD routine file</p> <ul style="list-style-type: none"> <li>• File name length—4-byte unsigned integer</li> <li>• File name—256-character (maximum) array</li> </ul>
Statement Fetch Begin	36	<p>Signifies that the Export operator is about to fetch the current statement in a request:</p> <ul style="list-style-type: none"> <li>• Statement number—4-byte unsigned integer</li> <li>• Block count—4-byte unsigned integer</li> </ul>
Statement Fetch End	37	<p>Signifies that the Export operator has fetched all the records for the current statement:</p> <ul style="list-style-type: none"> <li>• Record count—4-byte unsigned integer</li> </ul>
Request Fetch End	38	<p>Signifies that the Export operator has fetched all the records for the current request:</p> <ul style="list-style-type: none"> <li>• Records exported—4-byte unsigned integer</li> <li>• Records rejected—4-byte unsigned integer</li> </ul>
Export End	39	<p>Signifies that the Export operator has completed the export operation and displayed the number of exported records:</p> <ul style="list-style-type: none"> <li>• Records exported—4-byte unsigned integer</li> <li>• Records rejected—4-byte unsigned integer</li> </ul>
Block Count	40	<p>Signifies that the Export operator has determined the block count:</p> <ul style="list-style-type: none"> <li>• Block count—4-byte unsigned integer</li> </ul> <p>This event is only available when the NoSpool mode is used for the answer set.</p>
Statement Fetch End	41	<p>Signifies that the Export operator has fetched all the records for the current statement:</p> <ul style="list-style-type: none"> <li>• Record count—8-byte unsigned integer</li> </ul> <p>This event is only available when the value for the NotifyMethod attribute is set to 'Exit64' or 'ExitEON'.</p>
Request Fetch End	42	<p>Signifies that the Export operator has fetched all the records for the current request:</p> <ul style="list-style-type: none"> <li>• Records exported—8-byte unsigned integer</li> <li>• Records rejected—8-byte unsigned integer</li> </ul> <p>This event is only available when the value for the NotifyMethod attribute is set to 'Exit64' or 'ExitEON'.</p>
Export End	43	<p>Signifies that the Export operator has completed the export operation and displayed the number of exported records:</p> <ul style="list-style-type: none"> <li>• Records exported—8-byte unsigned integer</li> <li>• Records rejected—8-byte unsigned integer</li> </ul> <p>This event is only available when the value for the NotifyMethod attribute is set to 'Exit64' or 'ExitEON'.</p>
Request Submit End	44	<p>Signifies that the Export operator has received the response to the SELECT request:</p>

Event	Event Code	Event Description and Data Passed to the Notify Exit Routine
		<ul style="list-style-type: none"> <li>Statement count—4-byte unsigned integer</li> <li>Block count—8-byte unsigned integer</li> </ul> <p>This event is only available when the value for the NotifyMethod attribute is set to 'Exit64' or 'ExitEON'.</p>
Statement Fetch Begin	45	<p>Signifies that the Export operator is about to fetch the current statement in a request:</p> <ul style="list-style-type: none"> <li>Statement number—4-byte unsigned integer</li> <li>Block count—8-byte unsigned integer</li> </ul> <p>This event is only available when the value for the NotifyMethod attribute is set to 'Exit64' or 'ExitEON'.</p>
Block Count	46	<p>Signifies that the Export operator has determined the block count:</p> <ul style="list-style-type: none"> <li>Block count—8-byte unsigned integer</li> </ul> <p>This event is only available when the value for the NotifyMethod attribute is set to 'Exit64' or 'ExitEON'.</p>
Initialize	47	<p>Signifies successful processing of the notify feature:</p> <ul style="list-style-type: none"> <li>Version ID length—4-byte unsigned integer</li> <li>Version ID string—pointer to a null-terminated string</li> <li>Operator ID—4-byte unsigned integer</li> <li>Operator name length—4-byte unsigned integer</li> <li>Operator name string—pointer to a null-terminated string</li> <li>User name length—4-byte unsigned integer</li> <li>User name string—pointer to a null-terminated string</li> <li>Optional string length—4-byte unsigned integer</li> <li>Optional string—pointer to a null-terminated string</li> <li>Operator handle—4-byte unsigned integer</li> </ul> <p>This event is only available when the value for the NotifyMethod attribute is set to 'ExitEON'.</p>

The following table lists events that create notifications.

Event	Notification Level			Signifies
	Low	Medium	High	
Initialize	Yes	Yes	Yes	Successful processing of the notify option
Database Restart	No	Yes	Yes	A crash error from the database or CLlv2
CLlv2 Error	Yes	Yes	Yes	A CLlv2 error was encountered
Database Error	Yes	Yes	Yes	An error was encountered that terminates the Export operator
Exit	Yes	Yes	Yes	The Export operator is terminating
Export Begin	No	Yes	Yes	Opening the export file

Event	Notification Level			Signifies
	Low	Medium	High	
Request Submit Begin	No	Yes	Yes	Submitting the SELECT request
Request Submit End	No	Yes	Yes	Received SELECT request response
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

## Load Operator Events

The following table lists the event codes and describes the data that operators pass to the notify exit routine and the system log for each event.

**Note:**

To support future enhancements, always verify that notify exit routines *ignore* invalid or undefined event codes, and that they *do not* cause the operator to terminate abnormally.

Event	Event Code	Event Description and Data Passed to the Notify Exit Routine
Initialize	0	<p>Signifies successful processing of the notify feature:</p> <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>• Operator ID—4-byte unsigned integer</li> <li>• Operator name length—4-byte unsigned integer</li> <li>• Operator 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> <li>• Operator handle—4-byte unsigned integer</li> </ul>
Phase 1 Begin	2	<p>Signifies the beginning of the insert phase, where the table name is specified by the INSERT statement:</p> <ul style="list-style-type: none"> <li>• Table name length—4-byte unsigned integer</li> <li>• Table name—128-character (maximum) array</li> </ul>

Event	Event Code	Event Description and Data Passed to the Notify Exit Routine
Checkpoint	3	Signifies that checkpoint information is written to the restart log table: • Record number—4-byte unsigned integer
Phase 1 End	4	Signifies the CHECKPOINT LOADING END request has successfully completed after the end of the insert phase: • Records read—4-byte unsigned integer • Records sent to the database—4-byte unsigned integer
Phase 2 Begin	5	Signifies that the application phase is beginning: • No data accompanies the phase 2 begin event code
Phase 2 End	6	Signifies that the application phase is complete: • Records loaded—4-byte unsigned integer
Error Table 1	7	Signifies that processing of the SEL COUNT(*) request completed successfully for the first error table: • Number of rows—4-byte unsigned integer
Error Table 2	8	Signifies that processing of the SEL COUNT(*) request completed successfully for the second error table: • Number of rows—4-byte unsigned integer
Database Restart	9	Signifies that the Load operator received a crash message from the database or from the CLlv2: • No data accompanies the database restart event code
CLlv2 Error	10	Signifies that the Load operator received a CLlv2 error: • Error code—4-byte unsigned integer
Database Error	11	Signifies that the Load operator received an error that produces an Exit code of 12: • Error code—4-byte unsigned integer
Exit	12	Signifies that the Load operator is terminating: • Exit code—4-byte unsigned integer
Checkpoint	13	Signifies that checkpoint information is written to the restart log table: • Record number—8-byte unsigned integer  This event is only available when the value for the NotifyMethod attribute is set to 'Exit64' or 'ExitEON'.
Phase 1 End	14	Signifies the CHECKPOINT LOADING END request has successfully completed after the end of the insert phase: • Records read—8-byte unsigned integer • Records sent to the database—8-byte unsigned integer  This event is only available when the value for the NotifyMethod attribute is set to 'Exit64' or 'ExitEON'.
Phase 2 End	15	Signifies that the application phase is complete: • Records loaded—8-byte unsigned integer

Event	Event Code	Event Description and Data Passed to the Notify Exit Routine
		This event is only available when the value for the NotifyMethod attribute is set to 'Exit64' or 'ExitEON'.
Error Table 1	16	<p>Signifies that processing of the SEL CAST(COUNT(*) AS BIGINT) request completed successfully for the first error table:</p> <ul style="list-style-type: none"> <li>Number of rows—8-byte unsigned integer</li> </ul> <p>This event is only available when the value for the NotifyMethod attribute is set to 'Exit64' or 'ExitEON'.</p>
Error Table 2	17	<p>Signifies that processing of the SEL CAST(COUNT(*) AS BIGINT) request completed successfully for the second error table:</p> <ul style="list-style-type: none"> <li>Number of rows—8-byte unsigned integer</li> </ul> <p>This event is only available when the value for the NotifyMethod attribute is set to 'Exit64' or 'ExitEON'.</p>
Initialize	18	<p>Signifies successful processing of the notify feature:</p> <ul style="list-style-type: none"> <li>Version ID length—4-byte unsigned integer</li> <li>Version ID string—pointer to a null-terminated string</li> <li>Operator ID—4-byte unsigned integer</li> <li>Operator name length—4-byte unsigned integer</li> <li>Operator name string—pointer to a null-terminated string</li> <li>User name length—4-byte unsigned integer</li> <li>User name string—pointer to a null-terminated string</li> <li>Optional string length—4-byte unsigned integer</li> <li>Optional string—pointer to a null-terminated string</li> <li>Operator handle—4-byte unsigned integer</li> </ul> <p>This event is only available when the value for the NotifyMethod attribute is set to 'ExitEON'.</p>
Phase 1 Begin	19	<p>Signifies the beginning of the insert phase, where the table name is specified by the INSERT statement:</p> <ul style="list-style-type: none"> <li>Table name length—4-byte unsigned integer</li> <li>Table name—pointer to a null-terminated string</li> </ul> <p>This event is only available when the value for the NotifyMethod attribute is set to 'ExitEON'.</p>

The following table lists events that create notifications.

Event	Notification Level			Signifies
	Low	Med	High	
Initialize	Yes	Yes	Yes	Successful processing of the notify option
Phase 1 Begin	No	Yes	Yes	The acquisition phase is beginning
Checkpoint	No	No	Yes	Checkpoint information is written to the restart log table
Phase 1 End	No	Yes	Yes	Successful completion of the acquisition phase

Event	Notification Level			Signifies
	Low	Med	High	
Phase 2 Begin	No	Yes	Yes	The application phase is beginning
Phase 2 End	No	Yes	Yes	Successful completion of the application phase
Error Table 1	No	No	Yes	Successful processing of the SEL COUNT(*) request for the first error table
Error Table 2	No	No	Yes	Successful processing of the SEL COUNT(*) request for the second error table
Database Restart	No	Yes	Yes	A crash error was encountered from the database or CLlv2
CLlv2 Error	Yes	Yes	Yes	A CLlv2 error was encountered
Database Error	Yes	Yes	Yes	An error was encountered that will terminate the load operation
Exit	Yes	Yes	Yes	The Load operator is terminating

## Update Operator Events

The following table lists the event codes and describes the data that the Update operator passes to the notify exit routine for each event. The information in this table is also sent to the system log.

---

### Note:

To support future enhancements, always verify that your notify exit routines *ignore* invalid or undefined event codes, and that they *do not* cause the operator to terminate abnormally.

Event	Event Code	Event Description and Data Passed to the Notify Exit Routine
Initialize	0	Signifies successful processing of the notify feature: <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>• Operator ID—4-byte unsigned integer</li> <li>• Operator name length—4-byte unsigned integer</li> <li>• Operator 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> <li>• Operator handle—4-byte unsigned integer</li> </ul>
Phase 1 Begin	2	Signifies the beginning of the insert phase, where table name is specified by the DML statement:

Event	Event Code	Event Description and Data Passed to the Notify Exit Routine
		<ul style="list-style-type: none"> <li>• Table name length—4-byte unsigned integer</li> <li>• Table name—128-character (maximum) array</li> <li>• Table number—4-byte unsigned integer</li> </ul>
Checkpoint	3	Signifies that checkpoint information is written to the restart log table: <ul style="list-style-type: none"> <li>• Record number—4-byte unsigned integer</li> </ul>
Phase 1 End	4	Signifies the CHECKPOINT LOADING END request has successfully completed after the end of the acquisition phase: <ul style="list-style-type: none"> <li>• Records read—4-byte unsigned integer</li> <li>• Records skipped—4-byte unsigned integer</li> <li>• Records rejected—4-byte unsigned integer</li> <li>• Records sent to the database—4-byte unsigned integer</li> </ul>
Phase 2 Begin	5	Signifies that the application phase is beginning: <ul style="list-style-type: none"> <li>• No data accompanies the phase 2 begin event code</li> </ul>
Phase 2 End	6	Signifies that the application phase is complete. For each table in the request: <ul style="list-style-type: none"> <li>• Records inserted—4-byte unsigned integer</li> <li>• Records updated—4-byte unsigned integer</li> <li>• Records deleted—4-byte unsigned integer</li> <li>• Table number—4-byte unsigned integer</li> </ul>
Error Table 1	7	Signifies that processing of the SEL COUNT(*) request completed successfully for the first error table. <ul style="list-style-type: none"> <li>• Number of rows—4-byte unsigned integer</li> <li>• Table number—4-byte unsigned integer</li> </ul>
Error Table 2	8	Signifies that processing of the SEL COUNT(*) request completed successfully for the second error table. <ul style="list-style-type: none"> <li>• Number of rows—4-byte unsigned integer</li> <li>• Table number—4-byte unsigned integer</li> </ul>
Database Restart	9	Signifies that the Update operator received a crash message from the database or from CLlv2: <ul style="list-style-type: none"> <li>• No data accompanies the database restart event code</li> </ul>
CLLv2 Error	10	Signifies that the Update operator received a CLLv2 error: <ul style="list-style-type: none"> <li>• Error code—4-byte unsigned integer</li> </ul>
Database Error	11	Signifies that the Update operator received an error that will produce an Exit code of 12: <ul style="list-style-type: none"> <li>• Error code—4-byte unsigned integer</li> </ul> <p><b>Note:</b> Not all errors cause this event. A 3807 error, for example, while trying to drop or create a table does not terminate the Update operator.</p>
Exit	12	Signifies that the Update operator is terminating:

Event	Event Code	Event Description and Data Passed to the Notify Exit Routine
		<ul style="list-style-type: none"> <li>• Exit code—4-byte unsigned integer</li> </ul>
AMPs Down	21	<p>Signifies that the database has one or more non-operational AMPs, just prior to the acquisition phase:</p> <ul style="list-style-type: none"> <li>• No data accompanies the AMPs Down event code</li> </ul>
Import Begin	22	<p>Signifies that the first record is about to be read for each import task:</p> <ul style="list-style-type: none"> <li>• Import number—4-byte unsigned integer</li> </ul>
Import End	23	<p>Signifies that the last record is read for each import task. The returned data is the record statistics for the import task:</p> <ul style="list-style-type: none"> <li>• Records read—4-byte unsigned integer</li> <li>• Records skipped—4-byte unsigned integer</li> <li>• Records rejected—4-byte unsigned integer</li> <li>• Records sent to the database—4-byte unsigned integer</li> <li>• Import number—4-byte unsigned integer</li> </ul>
Delete Init	24	<p>Signifies successful processing of a DELETE statement:</p> <ul style="list-style-type: none"> <li>• No data accompanies the init. event code.</li> </ul>
Delete Begin	25	<p>Signifies that a DELETE statement is about to be sent to the database:</p> <ul style="list-style-type: none"> <li>• Table name length—4-byte unsigned integer</li> <li>• Table name—128-character (maximum) array</li> <li>• Table number—4-byte unsigned integer</li> </ul>
Delete End	26	<p>Signifies successful processing of a delete task:</p> <ul style="list-style-type: none"> <li>• Records deleted—4-byte unsigned integer</li> <li>• Table number—4-byte unsigned integer</li> </ul>
Delete Exit	27	<p>Signifies the end of a delete task:</p> <ul style="list-style-type: none"> <li>• Exit code—4-byte unsigned integer</li> </ul>
Checkpoint	28	<p>Signifies that checkpoint information is written to the restart log table:</p> <ul style="list-style-type: none"> <li>• Record number—8-byte unsigned integer</li> </ul> <p>This event is only available when the value for the NotifyMethod attribute is set to 'Exit64' or 'ExitEON'.</p>
Phase I End	29	<p>Signifies the CHECKPOINT LOADING END request has successfully completed after the end of the acquisition phase:</p> <ul style="list-style-type: none"> <li>• Records read—8-byte unsigned integer</li> <li>• Records skipped—8-byte unsigned integer</li> <li>• Records rejected—8-byte unsigned integer</li> <li>• Records sent to the database—8-byte unsigned integer</li> </ul> <p>This event is only available when the value for the NotifyMethod attribute is set to 'Exit64' or 'ExitEON'.</p>
Import End	30	<p>Signifies that the last record is read for each import task. The returned data is the record statistics for the import task:</p> <ul style="list-style-type: none"> <li>• Import number—4-byte unsigned integer</li> </ul>

Event	Event Code	Event Description and Data Passed to the Notify Exit Routine
		<ul style="list-style-type: none"> <li>• Records read—8-byte unsigned integer</li> <li>• Records skipped—8-byte unsigned integer</li> <li>• Records rejected—8-byte unsigned integer</li> <li>• Records sent to the database—8-byte unsigned integer</li> </ul> <p>This event is only available when the value for the NotifyMethod attribute is set to 'Exit64' or 'ExitEON'.</p>
Phase 2 End	31	Signifies that the application phase is complete for each table in the request: <ul style="list-style-type: none"> <li>• Table number—4-byte unsigned integer</li> <li>• Records inserted—8-byte unsigned integer</li> <li>• Records updated—8-byte unsigned integer</li> <li>• Records deleted—8-byte unsigned integer</li> </ul> <p>This event is only available when the value for the NotifyMethod attribute is set to 'Exit64' or 'ExitEON'.</p>
Error Table 1	32	Signifies that processing of the SEL CAST(COUNT(*) AS BIGINT) request completed successfully for the first error table: <ul style="list-style-type: none"> <li>• Table number—4-byte unsigned integer</li> <li>• Number of rows—8-byte unsigned integer</li> </ul> <p>This event is only available when the value for the NotifyMethod attribute is set to 'Exit64' or 'ExitEON'.</p>
Error Table 2	33	Signifies that processing of the SEL CAST(COUNT(*) AS BIGINT) request completed successfully for the first error table: <ul style="list-style-type: none"> <li>• Table number—4-byte unsigned integer</li> <li>• Number of rows—8-byte unsigned integer</li> </ul> <p>This event is only available when the value for the NotifyMethod attribute is set to 'Exit64' or 'ExitEON'.</p>
Initialize	34	Signifies successful processing of the notify feature: <ul style="list-style-type: none"> <li>• Version ID length—4-byte unsigned integer</li> <li>• Version ID string—pointer to a null-terminated string</li> <li>• Operator ID—4-byte unsigned integer</li> <li>• Operator name length—4-byte unsigned integer</li> <li>• Operator name string—pointer to a null-terminated string</li> <li>• User name length—4-byte unsigned integer</li> <li>• User name string—pointer to a null-terminated string</li> <li>• Optional string length—4-byte unsigned integer</li> <li>• Optional string—pointer to a null-terminated string</li> <li>• Operator handle—4-byte unsigned integer</li> </ul> <p>This event is only available when the value for the NotifyMethod attribute is set to 'ExitEON'.</p>
Phase 1 Begin	35	Signifies the beginning of the insert phase, where table name is specified by the DML statement: <ul style="list-style-type: none"> <li>• Table name length—4-byte unsigned integer</li> <li>• Table name—pointer to a null-terminated string</li> </ul>

Event	Event Code	Event Description and Data Passed to the Notify Exit Routine
		<ul style="list-style-type: none"> <li>• Table number—4-byte unsigned integer</li> </ul> <p>This event is only available when the value for the NotifyMethod attribute is set to 'ExitEON'.</p>
Delete Begin	36	<p>Signifies that a DELETE statement is about to be sent to the database:</p> <ul style="list-style-type: none"> <li>• Table name length—4-byte unsigned integer</li> <li>• Table name—pointer to a null-terminated string</li> <li>• Table number—4-byte unsigned integer</li> </ul> <p>This event is only available when the value for the NotifyMethod attribute is set to 'ExitEON'.</p>

The following table lists events that create notifications. Some events create notifications only for import tasks, some only for delete tasks, and some for both.

Event	Import Task	Delete Task	Notification Level			Signifies
			Low	Med	High	
Initialize	x		Yes	Yes	Yes	Successful processing of the notify option
Phase 1 Begin	x		No	Yes	Yes	The acquisition phase is beginning
Checkpoint	x		No	No	Yes	Checkpoint information is written to the restart log table
Phase 1 End	x		No	Yes	Yes	Successful completion of the acquisition phase
Phase 2 Begin	x	x	No	Yes	Yes	The application phase is beginning
Phase 2 End	x		No	Yes	Yes	Successful completion of the application phase
Error Table 1	x		No	No	Yes	Successful processing of the SEL COUNT(*) request for the first error table
Error Table 2	x	x	No	No	Yes	Successful processing of the SEL COUNT(*) request for the second error table
Database Restart	x	x	No	Yes	Yes	A crash error was encountered from the database or CLlv2 that will terminate the load operation
CLlv2 Error	x	x	Yes	Yes	Yes	A CLlv2 error was encountered
Database Error	x	x	Yes	Yes	Yes	An error was encountered that will terminate the load operation
Exit	x		Yes	Yes	Yes	The Update operator is terminating

Event	Import Task	Delete Task	Notification Level			Signifies
			Low	Med	High	
AMPs Down	x	x	No	No	Yes	Non-operational AMPs on the database
Import Begin	x		No	No	Yes	First record about to be read
Import End	x		No	No	Yes	Last record is read
Delete Init		x	Yes	Yes	Yes	The delete task is about to begin
Delete Begin		x	No	Yes	Yes	DELETE statement about to be sent to the database
Delete End		x	No	Yes	Yes	Successful delete task processing
Delete Exit		x	Yes	Yes	Yes	End of delete task

## Stream Operator Events

The following table lists the event codes and describes the data that the Stream operator passes to the notify exit routine or the system log.

**Note:**

To support future enhancements, always verify that your notify exit routines *ignore* invalid or undefined event codes, and that they *do not* cause the operator to terminate abnormally.

Event	Event Code	Event Description and Data Passed to the Notify Exit Routine
Initialize	0	<p>Signifies successful processing of the notify option:</p> <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>• Operator ID—4-byte unsigned integer</li> <li>• Operator name length—4-byte unsigned integer</li> <li>• Operator 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> <li>• Operator handle—4-byte unsigned integer</li> </ul>
Checkpoint Begin	2	<p>Signifies that the Stream operator is about to perform a checkpoint operation</p> <p>Record number—4-byte unsigned integer</p>
Import Begin	3	<p>Signifies that the first record is about to be read for each import task:</p> <ul style="list-style-type: none"> <li>• Import number—4-byte unsigned integer</li> </ul>

Event	Event Code	Event Description and Data Passed to the Notify Exit Routine
Import End	4	<p>Signifies that the last record is read for each import task. The returned data is the record statistics for the import task:</p> <ul style="list-style-type: none"> <li>• Import number—4-byte unsigned integer</li> <li>• Records read—4-byte unsigned integer</li> <li>• Records skipped—4-byte unsigned integer</li> <li>• Records rejected—4-byte unsigned integer</li> <li>• Records sent to the database—4-byte unsigned integer</li> <li>• Data Errors—4-byte unsigned integer.</li> </ul>
Error Table	5	<p>Signifies that processing of the SEL COUNT(*) request completed successfully for the error table:</p> <ul style="list-style-type: none"> <li>• Table Name—128-byte character (maximum) array.</li> <li>• Number of Rows—4-byte unsigned integer.</li> </ul>
Database Restart	6	<p>Signifies that the Stream operator received a crash message from the database or from CLIV2:</p> <ul style="list-style-type: none"> <li>• No data accompanies the database restart event code</li> </ul>
CLIV2 Error	7	<p>Signifies that the Stream operator received a CLIV2 error:</p> <ul style="list-style-type: none"> <li>• Error code—4-byte unsigned integer</li> </ul>
Database Error	8	<p>Signifies that the Stream operator received an error that will produce an Exit code of 12:</p> <ul style="list-style-type: none"> <li>• Error code—4-byte unsigned integer.</li> </ul> <p><b>Note:</b> Not all errors cause this event. A 3807 error, for example, while trying to drop or create a table, does not terminate the Stream operator.</p>
Exit	9	<p>Signifies that the Stream operator completed a load task:</p> <ul style="list-style-type: none"> <li>• Exit code—4-byte unsigned integer</li> </ul>
Table Statistics	10	<p>Signifies that the Stream operator has successfully written the table statistics:</p> <ul style="list-style-type: none"> <li>• Type (I = Insert, U = Update, D = Delete, or M = Merge) — 1-byte character variable.</li> <li>• Database Name — 64-character (maximum) array.</li> <li>• Table/Macro Name — 64-character (maximum) array</li> <li>• Activity count—4-byte unsigned integer</li> </ul>
Checkpoint End	11	<p>Signifies that the Stream operator has successfully completed the checkpoint operation:</p> <ul style="list-style-type: none"> <li>• Record number—4-byte unsigned integer</li> </ul>
Interim Run Statistics	12	<p>Signifies that the Stream operator is flushing the stale buffers (because a latency interval has expired), has just completed a checkpoint, or has read the last record for an import task. The returned data is the statistics for the current load.</p> <ul style="list-style-type: none"> <li>• Import number—4-byte unsigned integer</li> <li>• Statements sent to the database—4-byte unsigned integer</li> <li>• Requests sent to the database—4-byte unsigned integer</li> </ul>

Event	Event Code	Event Description and Data Passed to the Notify Exit Routine
		<ul style="list-style-type: none"> <li>Records read—4-byte unsigned integer</li> <li>Records skipped—4-byte unsigned integer</li> <li>Records rejected—4-byte unsigned integer</li> <li>Records sent to the database—4-byte unsigned integer</li> <li>Data errors—4-byte unsigned integer</li> </ul>
DML Error	13	<p>Signifies that the Stream operator received an error that was caused by DML and will introduce an error-row insert to the error table.</p> <ul style="list-style-type: none"> <li>Import number—4-byte unsigned integer</li> <li>Error code—4-byte unsigned integer</li> <li>Error message—256-character (maximum) array</li> <li>Record number—4-byte unsigned integer</li> <li>Data Input number—1-byte unsigned char</li> <li>DML number—1-byte unsigned char</li> <li>Statement number—1-byte unsigned char</li> <li>Record data—6,004-character (maximum) array</li> <li>Record data length—4-byte unsigned integer</li> <li>Feedback—a pointer to 4-byte unsigned integer</li> </ul> <p>“Feedback” always points to integer 0 when it is passed to the notify exit routine. You may change the value of this integer to 1 to instruct the Stream operator not to log the error to the error table. In this case, the Stream operator will not log the error, but will continue other regular processes on this error.</p>
Checkpoint Begin	14	<p>Signifies that the Stream operator is about to perform a checkpoint operation:</p> <ul style="list-style-type: none"> <li>Record number—8-byte unsigned integer</li> </ul> <p>This event is only available when the value for the NotifyMethod attribute is set to 'Exit64' or 'ExitEON'.</p>
Checkpoint End	15	<p>Signifies that the Stream operator has successfully completed the checkpoint operation:</p> <ul style="list-style-type: none"> <li>Record number—8-byte unsigned integer</li> </ul> <p>This event is only available when the value for the NotifyMethod attribute is set to 'Exit64' or 'ExitEON'.</p>
Error Table	16	<p>Signifies that processing of the SEL CAST(COUNT(*) AS BIGINT) request completed successfully for the error table:</p> <ul style="list-style-type: none"> <li>Table Name—128 byte character (maximum) array</li> <li>Number of Rows—8-byte unsigned integer</li> </ul> <p>This event is only available when the value for the NotifyMethod attribute is set to 'Exit64' or 'ExitEON'.</p>
Interim Run Statistics	17	<p>Signifies that the Stream operator is flushing the stale buffers (because a latency interval has expired) and has just completed a checkpoint or has read the last record for an import task. The returned data is the statistics for the current load:</p> <ul style="list-style-type: none"> <li>Import number—4-byte unsigned integer</li> <li>Statements sent to the database—8-byte unsigned integer</li> <li>Requests sent to the database—8-byte unsigned integer</li> </ul>

Event	Event Code	Event Description and Data Passed to the Notify Exit Routine
		<ul style="list-style-type: none"> <li>• Records read—8-byte unsigned integer</li> <li>• Records skipped—8-byte unsigned integer</li> <li>• Records rejected—8-byte unsigned integer</li> <li>• Records sent to the database—8-byte unsigned integer</li> <li>• Data errors—8-byte unsigned integer</li> </ul> <p>This event is only available when the value for the NotifyMethod attribute is set to 'Exit64' or 'ExitEON'.</p>
Import End	18	Signifies that the last record is read for each import task. The returned data is the record statistics for the import task: <ul style="list-style-type: none"> <li>• Import number—4-byte unsigned integer</li> <li>• Records read—8-byte unsigned integer</li> <li>• Records skipped—8-byte unsigned integer</li> <li>• Records rejected—8-byte unsigned integer</li> <li>• Records sent to the database—8-byte unsigned integer</li> <li>• Data Errors—8-byte unsigned integer.</li> </ul> <p>This event is only available when the value for the NotifyMethod attribute is set to 'Exit64' or 'ExitEON'.</p>
Table Statistics	19	Signifies that the Stream operator has successfully written the table statistics: <ul style="list-style-type: none"> <li>• Type (I = Insert, U = Update, D = Delete, or M = Merge) — 1-byte character variable.</li> <li>• Database Name — pointer to a null-terminated string.</li> <li>• Table/Macro Name — pointer to a null-terminated string</li> <li>• Activity count—8-byte unsigned integer</li> </ul> <p>This event is only available when the value for the NotifyMethod attribute is set to 'Exit64' or 'ExitEON'.</p>
Initialize	20	Signifies successful processing of the notify feature: <ul style="list-style-type: none"> <li>• Version ID length—4-byte unsigned integer</li> <li>• Version ID string—pointer to a null-terminated string</li> <li>• Operator ID—4-byte unsigned integer</li> <li>• Operator name length—4-byte unsigned integer</li> <li>• Operator name string—pointer to a null-terminated string</li> <li>• User name length—4-byte unsigned integer</li> <li>• User name string—pointer to a null-terminated string</li> <li>• Optional string length—4-byte unsigned integer</li> <li>• Optional string—pointer to a null-terminated string</li> <li>• Operator handle—4-byte unsigned integer</li> </ul> <p>This event is only available when the value for the NotifyMethod attribute is set to 'ExitEON'.</p>

The following table lists events that create notifications.

Event	Notification Level				Signifies
	Low	Medium	High	Ultra	
Initialize	Yes	Yes	Yes	Yes	Successful processing of the notify option)
Checkpoint Begin	No	No	Yes	Yes	Stream operator started a checkpoint
Import Begin	No	No	Yes	Yes	Stream operator is about to start reading records
Import End	No	No	Yes	Yes	Last record is read
Error Table	No	No	Yes	Yes	Successful processing of the SEL COUNT(*) request for the error table
Database Restart	No	Yes	Yes	Yes	Stream operator received a crash error from the database or CLIV2
CLIV2 Error	Yes	Yes	Yes	Yes	A CLIV2 error was encountered
Database Error	Yes	Yes	Yes	Yes	A crash error was encountered from the database or CLIV2 that will terminate the load operation
Exit	Yes	Yes	Yes	Yes	Stream operator is terminating
Table Statistics	No	Yes	Yes	Yes	Stream operator has successfully written the table statistics
Checkpoint End	No	No	Yes	Yes	Stream operator successfully completed a checkpoint
Interim Run Statistics	No	No	No	Yes	Stream operator is about to flush the stale buffers (because a latency interval has expired), or Stream operator successfully completed a checkpoint, or an Import is just successfully completed
DML Error	No	No	Yes	Yes	Stream operator is about to log a DML error to the error table

## DataConnector Operator Events

The following table lists the event codes and describes the data that the DataConnector operator passes to the notify exit routine for each event. The information in this table is also sent to the system log.

### Note:

To support future enhancements, always verify that your notify exit routines ignore invalid or undefined event codes, and that they do not cause the operator to terminate abnormally.

Event Code	Notification Level				Event Description
	Event	Low	Med	High	
0	Initialize	Yes	Yes	Yes	<p>Signifies successful processing of the notify feature.</p> <ul style="list-style-type: none"> <li>• Operator handle - 4-byte unsigned integer</li> <li>• Operator number - 4-byte unsigned integer</li> <li>• Operator count - 4-byte unsigned integer</li> <li>• Utility ID - 4 byte unsigned integer</li> <li>• Task ID - 32-character (maximum) array</li> <li>• Time stamp - 26-character (fixed size) array with the format YYYY-MM-DDbHH:MM:SS.SSSSSS</li> </ul> <p>The following offset fields contain the byte offsets to data items in the event record that are null terminated strings. Given the address of the event record, the address of each data item's string can be obtained from its offset.</p> <ul style="list-style-type: none"> <li>• Job ID offset - 4-byte unsigned integer</li> <li>• Version ID offset - 4-byte unsigned integer</li> <li>• Utility name offset - 4-byte unsigned integer</li> <li>• User string offset - 4-byte unsigned integer</li> </ul>
1	Directory Scan Complete	No	No	Yes	<p>Signifies successful processing of a scan for file names in a directory.</p> <ul style="list-style-type: none"> <li>• Operator number - 4-byte unsigned integer</li> <li>• Task name - 32-character (maximum) array</li> <li>• File count - 4-byte unsigned integer</li> <li>• Time stamp - 26-character (fixed size) array with the format YYYY-MM-DDbHH:MM:SS.SSSSSS</li> </ul> <p>The following offset fields contain the byte offsets to data items in the event record that are null terminated strings. Given the address of the event record, the address of such a string can be obtained from the offset.</p> <ul style="list-style-type: none"> <li>• Script name offset - 4-byte unsigned integer</li> <li>• Job name offset - 4-byte unsigned integer</li> <li>• Job ID offset - 4-byte unsigned integer</li> <li>• File list offset - 4-byte unsigned integer (Note: there are "File Count" NULL terminated file name strings arranged end-to-end in the file list)</li> </ul>
2	File Read Complete	No	No	Yes	<p>Signifies successful processing of a file in the file list.</p> <ul style="list-style-type: none"> <li>• Operator number - 4-byte unsigned integer</li> <li>• Task ID - 32-character (maximum) array</li> <li>• Record count - 4-byte unsigned integer</li> <li>• File size - 4-byte unsigned integer</li> </ul> <p>The following offset fields contain the byte offsets to data items in the event record that are null terminated strings. Given the address of the event record, the address of such a string can be obtained from the offset.</p> <ul style="list-style-type: none"> <li>• Script name offset - 4-byte unsigned integer</li> <li>• Job name offset - 4-byte unsigned integer</li> </ul>

Event Code	Event	Notification Level			Event Description
		Low	Med	High	
					<ul style="list-style-type: none"> <li>• Job ID offset - 4-byte unsigned integer</li> <li>• File name offset - 4-byte unsigned integer</li> </ul>
3	Checkpoint	No	No	Yes	<p>Signifies successful processing of a checkpoint.</p> <ul style="list-style-type: none"> <li>• Operator number - 4-byte unsigned integer</li> <li>• Task ID - 32-character (maximum) array</li> </ul> <p>The following offset fields contain the byte offsets to data items in the event record that are null terminated strings. Given the address of the event record, the address of such a string can be obtained from the offset.</p> <ul style="list-style-type: none"> <li>• Script name offset - 4-byte unsigned integer</li> <li>• Job name offset - 4-byte unsigned integer</li> <li>• Job ID offset - 4-byte unsigned integer</li> </ul>
4	Terminate	Yes	Yes	Yes	<p>Signifies successful termination of the notify feature.</p> <ul style="list-style-type: none"> <li>• Operator number - 4-byte unsigned integer</li> <li>• Task ID - 32-character (maximum) array</li> <li>• Return code - 4-byte unsigned integer</li> <li>• Time stamp - 26-character (fixed size) array with the format YYYY-MM-DDHH:MM:SS.SSSSSS</li> </ul>

## Using Notify Exit Routines to Monitor Events

A notify exit routine (sometimes called an INMOD) specifies a predefined action to be performed whenever certain significant events occur during a Teradata PT job. Notify exit routines are especially useful in environments where job scheduling relies heavily on automation to optimize system performance.

For example, by writing an exit routine in C (without using CLIV2) and using the NotifyMethod and NotifyExit attributes, 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. In all cases, notify exit routines are dynamically loaded at run time, rather than link edited into the Teradata PT module.

The entry point for notify exit routines for all notify exits must be `_dynamn`.

## Interface

Teradata PT accumulates operational information about specific events that occur during a job. If the script includes a notify attribute with an exit option specification, then, when the specific events occur, Teradata PT calls the named notify exit routine and passes to it:

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

The event codes and the data that Teradata PT passes to the notify exit routine for each event encountered are described in the individual operators. The events associated with each level of notification (low, medium and high) are also described in the individual operators.

**Note:**

To support future enhancements, always make sure that your notify exit routines *ignore* invalid or undefined event codes, and that they *do not* cause Teradata PT to terminate abnormally.

## Samples

Teradata PT includes sample notify exit routines that show you how to write a notify exit routine using the C programming language.

Sample notify exit routines for the following operators are installed in the sample folder:

- Load operator (`ldnfyext.c`)
- Update operator (`updnyfxt.c`)
- Export operator (`expnfyxt.c`)
- Stream operator (`stmnfyxt.c`)
- Data Connector (`dcrnfyxt.c`)

The IBM z/OS samples are installed in SAMPLIB:

**z/OS Notify Exit Names**

Description	z/OS Name
Load Operator	PTXLDNTO
Update Operator	PTXUPNTO
Export Operator	PTXEPNTO
Stream Operator	PTXSTNTO
Data Connector Operator	PTXDCNTO

## Compiling and Linking Notify Exit Routines

To meet your particular system needs, you can either write your own notify exit routine, or you can modify the sample notify exit routine provided with your Teradata PT software.

Whenever you create or modify exit routines, you must compile the routine and link it into a shared object for use by Teradata PT.

## UNIX Systems

## Solaris Running on a SPARC System

Use the following syntax to compile source files into a shared object module for notify exit routines on Solaris running on SPARC systems:

```
cc -G — -KPIC [sourcefile] -o shared-object-name ➔
```

where:

### UNIX Operating System Syntax for Notify Exit Routines

Syntax Element	Description
<i>cc</i>	Call to the program that invokes the native UNIX C compiler.
<i>-G</i>	Linker option that generates a shared object file.
<i>-KPIC</i>	Compiler option that generates Position Independent Code (PIC) for all notify exit routines.
<i>-o</i>	Switch to the linker.
<i>shared-object-name</i>	Name of your shared object file. The <i>shared-object-name</i> can be any valid UNIX file name. This is the name you specify in the NotifyExit attribute value supplied in the operator definition section of a job script. <b>Note:</b> When creating a shared object module for a notify exit routine, if the notify exit routine uses functions from an external library, then that library must be statically linked with the notify exit routine so that Teradata PT can resolve the external references.
<i>sourcefile</i>	UNIX file name(s) of the source file(s) for your notify exit routine.

## Linux

Use the following syntax to compile source files into a shared object module for notify exit routines on Linux client systems:

```
gcc [ -shared sourcefile ] -o shared-object-name ➔
```

where:

### Linux Syntax for Notify Exit Routines

Syntax Element	Description
<code>gcc</code>	Call to the program that invokes the gcc compiler.
<code>-shared</code>	Link option that generates a shared object file.
<code>-o</code>	Switch to the link option.
<code>shared-object-name</code>	<p>Any valid file name to serve as the name of the shared object file. Specify this name in the NotifyExit attribute value supplied in the operator definition section of a job script.</p> <p><b>Note:</b> When creating a shared object module for a notify exit routine, if the notify exit routine uses functions from an external library, then that library must be statically linked with the notify exit routine so that Teradata PT can resolve the external references.</p>
<code>sourcefile</code>	File name(s) of the source file(s) for a notify exit routine.

## IBM AIX

Use the following syntax to compile and link source files into a shared object module for notify exit routines on IBM AIX client systems:

### Compile Syntax

```
cc — -c — -brtl — -fPIC — [sourcefile.c] —————→|
```

### Link Syntax

```
ld -G — -e_dynamn — -bE:export_dynamn.txt —————→|  
(A) [objectfile.o] — -o shared-object-name — -lm — -lc —————→|
```

where:

### IBM AIX Syntax for Notify Exit Routines

Syntax Element	Description
<code>-be:export_dynamn.txt</code>	Linker option that exports the symbol "_dynamn" explicitly and the file export-dynamn.txt contains the symbol.

Syntax Element	Description
-bexpall	Option that exports all global symbols, except imported symbols, un-referenced symbols defined in archive members, and symbols beginning with an underscore (_).
-brtl	Option that tells the linkage editor to accept both .so and .a library file types.
-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.
-e_dynamn	Option that sets the entry point of the notify exit routine to _dynamn.
-fPIC	Compiler option that generates Position Independent Code for all notify exit routines.
-G	Option that produces a shared object-enabled for use with the runtime linker.
-lc	Link with the /lib/libc.a library.
ld	Call to the program that invokes the native UNIX linker.
-lm	Link with the /lib/libc.a library.
-o	Switch to the linker.
objectfile	File that the compiler generates and linker uses to generate shared-object-name.
shared-object-name	The <i>shared-object-name</i> can be any valid UNIX file name. This is the name you specify in the NotifyExit attribute value supplied in the operator definition section of a job script. When creating a shared object module for a notify exit routine, if the notify exit routine uses functions from an external library, then that library must be statically linked with the notify exit routine so that Teradata PT can resolve the external references.
sourcefile	UNIX file name(s) of the source file(s) for your notify exit routine.

## IBM OS z/OS

Use:

- JCL member PT\$NTFYX in SAMPLIB to compile and link the Notify Exits, or
- The following syntax from the shell or OMVS to compile and link source files to a DLL:

```
cc -o "'MVS.PDSE.LOAD(module_name)'"
      "'MVS.PDSE.C(source_name.c)'"
      -W c,dll,expo -W l,dll
```

where:

**z/OS Syntax for Notify Exit Routines**

Syntax Element	Description
<b>-o</b>	Name of the executable load module that is produced during the link-edit phase. In the following example, a load module named INMOD is placed in a z/OS PDSE named PDSE.LOAD, and the source code, INMOD, is compiled as a member of a z/OS PDSE named TEST.C:  <pre>cc -o //'PDSE.LOAD(INMOD)' " //'"TEST.C(INMOD)' " -W c,dll,expo -W l,dll</pre> Member names are limited to eight characters.
<b>-W c,dll,expo</b>	Options passed to the compiler phase to indicate that the source is to be compiled as a dll with all functions exported.
<b>-W l,dll</b>	Options passed to the link-edit phase to indicate that the module are linked as a dll.

## Windows Platforms

Consider the following:

- To generate and use a notify exit routine with Teradata PT on a Windows client system, the routine must meet the following requirements:
  - Written in C
  - The `_dynamn` entry point is `__declspec`
  - Saved as a dynamic-link library (DLL) file
- To generate an INMOD or notify exit routine on a workstation-attached Windows client, do the following:
  1. Edit the routine source file and ensure that the `dynamn` is named "`__declspec`."
  2. Create a dynamic link library by entering the following command (in Microsoft C compiler syntax) where `sourcefilename` is the name of your INMOD or notify exit routine source file:  
`c1 /DWIN32 /LD sourcefilename`  
 This command produces a file with the same name as the source file, but with a .dll file extension.
- To use a compiler other than the Microsoft C compiler the documentation for that compiler for instructions on creating .dll files.

# Advanced Database Considerations

## Query Banding Considerations

Available through the Load, Update, Stream, Export, SQL Selector, SQL Inserter and DDL operators, use the QueryBandSessInfo attribute to set the query banding feature of Vantage.

The QueryBandSessInfo attribute may be specified as an ARRAY attribute so that operators for which the QueryBandSessInfo attribute is available can send a separate SET QUERY\_BAND request to the database for each entry of query band data in the array attribute.

There are two interaction items to consider for systems that are using Teradata Dynamic Workload Manager (Teradata DWM) and Teradata PT:

- Teradata Dynamic Workload Manager classifies the first utility-specific command in for a Teradata PT operator. Classification is based on query bands if the QueryBandSessInfo attribute contains one or more values.
- When the QueryBandSessInfo attribute contains one or more values, the system sets a flag in the session context so that all subsequent operator commands are not classified. This ensures that all work for the operator runs at the same priority and workload definition.
- When the Delay option is specified, the TENACITY/TenacityHours and SLEEP/TenacitySleep attributes become ineffective for that job because Teradata DWM will automatically delay a logon until it is eligible to run.

## Large Objects

Teradata PT supports loading Large Object (LOB) data into, and extracting LOB data from, database tables.

Large Objects are data types that can be Binary Large Objects (BLOBs) or Character Large Objects (CLOBs).

Depending on how the LOB columns are defined in the Teradata PT schema, Teradata PT uses one of the following two methods:

- Inline
- Deferred

to load LOB data into, or extract LOB data from, the database.

### Note:

- Teradata PT supports the inline mode on all platforms.
- Teradata PT supports the deferred mode on all platforms, except for the z/OS platform, because data sets are not assigned to the LOB files.

## Defining LOB Data in a Teradata PT Schema

The syntax for defining LOB data, either as BLOB or CLOB, in a Teradata PT schema is shown here:

- **BLOB (*lengthBytes*) or CLOB (*lengthBytes*)**

This syntax indicates that the Teradata PT load or extract job will use the inline method to transfer the LOB data between the Teradata PT and the database.

- **BLOB (*lengthBytes*) AS DEFERRED BY NAME or CLOB (*lengthBytes*) AS DEFERRED BY NAME**

This syntax indicates that the LOB data is transferred using the deferred method.

When LOB columns are defined as DEFERRED BY NAME in the schema, Teradata PT expects regular VARCHARs in place of deferred LOB columns in a data row. Each VARCHAR is a file name of a flat file that contains the actual data for a deferred LOB column. Teradata PT refers to these VARCHARs as LOB file locators.

### Inline Method

In the inline method, data rows sent across the Teradata PT data stream must contain both non-LOB and LOB data. The inline BLOB and CLOB columns in the data rows are treated as similar as the VARBYTE and VARCHAR columns.

However, these BLOB and CLOB columns must have an 8-byte integer as their length indicator field instead of 2-byte length indicator as in the VARBYTE and VARCHAR columns.

One restriction in using the inline method to transfer LOB data is that the entire data row should not be larger than 1 MB since that is the row-size limit imposed by the Teradata PT data stream.

### Deferred Method

In the deferred method, the Teradata PT does not send the actual LOB data across its data stream for LOB columns defined as DEFERRED BY NAME.

Deferred LOB data is processed entirely separately from the non-LOB data. Data rows sent across the Teradata PT data stream contain only non-LOB data and LOB file locators. The locators point to the flat files containing actual data for the deferred LOB columns.

## Moving LOB Data from one Database Table to Another

To move LOB data from one database table to another, a Teradata PT script must use:

- The SQL Selector operator to extract LOB data from a source table, and
- The SQL Inserter, Update, or Stream operator to load the data to a target table.

## Using the Inline Method

If you want to use the inline method for the job, you must define the LOB columns as BLOB (*lengthBytes*) or CLOB (*lengthBytes*), according to their data content, in the schema.

## Using the Deferred Method

If you want to use the deferred method for the job, you must define the LOB columns as BLOB (*lengthBytes*) AS DEFERRED BY NAME or CLOB (*lengthBytes*) AS DEFERRED BY NAME, according to their data content, in the schema.

During the job run, be aware of the following events:

- The SQL Selector operator retrieves data from deferred LOB columns and writes the data to external data files. These data files reside temporarily under the directory specified in LobDirectoryPath attribute or under the current working directory if the LobDirectoryPath attribute is not used.
- One data file represents data for one LOB column.
- These LOB data files have formulaic names as follows:

`<file-basename>_<column-name>_c#<job-id>_p<#>_r<#>.<file-extension>`

where:

- `<file-basename>` is the value specified in the LobFileBaseName attribute.
- `<column-name>` is the column name defined in a source table.
- `<job-id>` is the job name from the **tbuild** command line.
- `<#>` is an integer number generated internally by Teradata PT
- `<file-extension>` is the value specified in the LobFileExtension attribute.

These LOB data files need to be deleted immediately after their data has been inserted into the target table.

For a sample Teradata PT script in the `<install-directory>/sample/userguide` directory, see `PST00021.txt`.

## Loading LOB Data into a Database Table from External Data Files

To load LOB data into a database table from external data files, the Teradata PT script must use:

- The DataConnector producer operator (to read LOB data from an external data file)
- The Inserter operator (to insert LOB data to a target table)

If the LOB columns are defined as BLOB (*lengthBytes*) or CLOB (*lengthBytes*) in the schema, data records in the external data file specified in the FileName attribute must have an 8-byte integer length indicator preceding the BLOB or CLOB data.

If the LOB columns are defined as BLOB (*lengthBytes*) AS DEFERRED BY NAME or CLOB (*lengthBytes*) AS DEFERRED BE NAME, data records in the external file specified in FileName attribute must have regular VARCHARs, known as LOB file locators, where the LOB columns are defined as deferred.

For a sample Teradata PT script in the <install-directory>/sample/userguide directory, see PST00006.txt.

## Extracting LOB Data from Database Tables and Writing LOB Data to an External Target

To extract LOB data from database tables and write LOB data to an external target, the Teradata PT script must use:

- The SQL Selector operator (to extract LOB data from a source table)
- The DataConnector operator (to write on an external target)

The LOB columns must be defined accordingly to their data content as BLOB (*lengthBytes*) AS DEFERRED BY NAME or CLOB (*lengthBytes*) AS DEFERRED BY NAME in the schema.

The job produces the following outcomes:

- LOB data files located under the directory specified in the LobDirectoryPath attribute or under the current working directory if there is no value specified for the LobDirectoryPath attribute. For the file name formula, see [Using the Deferred Method](#).
- An external data file whose name is specified in the FileName attribute. Data records written to the file contain non-LOB data and LOB file locators that identify the files containing the LOB data. These LOB file locators have the same format as VARCHAR columns and contain the file names of the LOB data files produced by the SQL Selector operator.

For a sample Teradata PT script in the <install-directory>/sample/userguide directory, see PST00021.txt.

## Moving LOB Data from an External Database Table to a Database Table

To move LOB data from an external database table to a database, a Teradata PT script must use the following:

- The ODBC operator to extract LOB data from a source table
- The SQL Inserter, Update, or Stream operator to load the data to a target table

## Using the Inline Method

To use the inline method for a job, you must define the LOB columns as BLOB (*lengthBytes*) or CLOB (*lengthBytes*), according to their data content, in the schema

## Using the Deferred Method

To use the deferred method for the job, you must define the LOB columns as BLOB (*lengthBytes*) AS DEFERRED BY NAME or CLOB (*lengthBytes*) AS DEFERRED BY NAME, according to their data content, in the schema.

You should be aware of the following events during the job run:

- The ODBC operator retrieves data from deferred LOB columns and writes the data to external data files. These data files reside temporarily under the directory specified in LobDirectoryPath attribute or under the current working directory if the LobDirectoryPath attribute is not used.
- One data file represents data for one LOB column.

These LOB data files have formulaic names as follows:

```
<file-basename>_<column-name>_<job-id>_p<#>_r<#>.<file-extension>
```

where:

- <file-basename> is the value specified in the LobFileBaseName attribute.
- <column-name> is the column name defined in a source table.
- <job-id> is the job name from the tbuild command line.
- <#> is an integer number generated internally by Teradata PT.
- <file-extension> is the value specified in the LobFileExtension attribute.

These LOB data files will be deleted immediately after their data has been inserted into the target table.

## User Considerations

- If possible, it is recommended to use the inline method to transfer LOB data between Teradata PT and the database because this method gives better performance, especially when multiple instances of the Inserter operator are used in the loading job.

However, there is a drawback in using the inline method, which is that the entire data row cannot be larger than 1000000 bytes, which is the current row-size limit imposed by the database.

- When a massive amount of data (may be 1,000,000 rows or more) is loaded in a database table and the data row size is not larger than 1000000 bytes, the user may consider loading BLOBS as VARBYTES and CLOBS as VARCHAR. The database can perform the trivial data conversion of VARBYTE to BLOB and VARCHAR to CLOB. That allows Teradata PT to use the Export operator to extract LOB data (defined to it as VARBYTE or VARCHAR columns) and the Load or Update operator to load LOB data (defined to it as VARBYTE or VARCHAR columns) at high speed.
- If you want to load BLOB or CLOB data into a table as VARBYTE or VARCHAR, the following needs to be done in a Teradata PT job script:
  - BLOB columns must be defined as VARBYTE.

- CLOB columns must be defined as VARCHAR.
- Use the CAST function in the SQL SELECT statement specified in the SelectStmt attribute to explicitly convert BLOB to VARBYTE and CLOB to VARCHAR if data is extracted from a database table.
- When the SQL Selector operator extracts deferred LOB data from a database table and the SQL Inserter loads it into another database table, the LOB data files associated with each row are deleted after the row has been inserted successfully. By the end of the loading job, all LOB data files that the SQL Selector operator created are deleted.
- When a Teradata PT job extracts LOB data from a database table using the deferred method and writes it to an external target, the LOB data files associated with each record written to the external target will not be deleted at the end of the job.
- For a job loading LOB data into a database table from an external file (non-database source), a user can define both inline and deferred LOB data types in the same schema. The job script uses the DataConnector operator as producer and Inserter operator as consumer. Thus, the following schema is valid for the job:

```
col1 CLOB(2000),
col2 BLOB(2000),
col3 CLOB(75000) AS DEFERRED BY NAME,
col4 BLOB(75000) AS DEFERRED BY NAME
```

---

**Note:**

If the job script reads LOB data from a database table, the mentioned schema is not valid. A schema for this extraction scenario cannot contain both inline and deferred LOB columns.

---

## Limitations on LOB Processing

- Teradata PT does not support the use of multiple APPLY when deferred LOB data types are defined in a job schema and the LOB data is moved from one database table to one or more target table simultaneously.
- Teradata PT does not support writing LOB data to a z/OS dataset defined by a JCL DD statement. This means that the LOB exporting feature can only work with HFS files on z/OS platforms. Users cannot specify a z/OS PDS name for the location where the LOB data files are written or for the file names.
- Teradata PT does not allow a LOB column to be used in any predicate evaluation specified in the WHERE clause or in either type of CASE expression.
- If LOB data sent to the database is larger than the size of the LOB column defined in the schema of the target table, the database truncates the data. This behavior is consistent with the behavior of VARCHAR, VARBYTE or VARGRAPHIC column types. But Teradata PT checks the size of a LOB data file against its schema to prevent the database from truncating the data with no user warning.

Teradata PT checks if the LOB data file is a flat file. If it is, the SQL Inserter operator compares the file size with the LOB column size defined in the Teradata PT schema. If the file size is bigger, then the job is terminated with a detailed error message.

If the LOB data source is not a file (it can be a named pipe, for instance), the SQL Inserter operator cannot make the file size check, and thus, if the data size exceeds the defined column size, the database truncates the data without any warning message.

## JavaScript Object Notation

Teradata PT supports loading JavaScript Object Notation (JSON) data into, and extracting JSON data from, database tables.

The wire protocol between Teradata PT and the database treats JSON as a CLOB.

Depending on how the JSON columns are defined in the Teradata PT schema, Teradata PT uses one of the following two methods to load JSON data into, or extract JSON data from, the database:

- Inline
- Deferred

The JSON columns in the Teradata PT schema must have a length in bytes.

Only the TPT SQL Selector operator can extract JSON data from a database table.

Only the TPT SQL Inserter, Update, and Stream operators can load JSON data into a database table.

Only the TPT DataConnector operator can read JSON data from, and write JOS data to, a file.

For more information on CLOB, see [Large Objects](#).

## AVRO Data Support

Teradata PT supports loading AVRO data into, and extracting AVRO data from, database tables.

The wire protocol between Teradata PT and the database treats AVRO as a VARBYTE/BLOB (depending on the size of the data).

Teradata PT does not have an AVRO data type for specification in the schema.

The user must use VARBYTE or BLOB in the schema definition, and all of the rules for loading AVRO data into the database (or extracting from the database) are the same as how LOBs are supported.

This means that inline and deferred mode is allowed.

The database has implicit CASTs between VARBYTE/BLOB and AVRO.

The maximum size allowed for an AVRO container is 2GB (the same as the maximum size of a BLOB column in the database). If the AVRO container is larger than 2GB, the container must be split into multiple, smaller containers prior to loading into the database.

For more information on BLOB, see [Large Objects](#).

## Enhanced LOB Support

Teradata PT now supports schemas containing rows with inline LOBs (CLOB, BLOB, JSON, and XML data types) that exceed the 1MB record limit. Non-LOB data and smaller inline LOBs (total not to exceed 1MB) will be transferred as is. But larger inline LOB columns will be transferred in deferred mode. This is done automatically by the operators without any preparation by the user. Temporary LOB files will be created by the producer operator. Once processed by the consumer operator, the temporary LOB files will be removed.

Therefore, large LOB columns do not have to be transferred using the "AS DEFERRED BY NAME" clause and pre-defined LOB files do not need to exist.

### Important Details

- Teradata recommends defining a CLOB, BLOB, or JSON column using a specific size. If no size is specified, then the maximum size is used, making the transfer less efficient.
- Define the sizes for CLOB, BLOB, and JSON columns as accurately as possible. For example, if the maximum size for one particular object is 12,000 bytes, but a definition of CLOB(50000) is used, that leads to more inefficiency.
- Ensure that there is enough disk space for Teradata PT to create all the temporary LOB files that it needs to create. These temporary LOB files will be removed, but disk usage may be high during the transfer.
- Ensure that there is enough memory available in the system whenever transferring large LOB columns. Both the producer and the consumer have to store the LOB data in memory at some point.
- If an active job is killed, temporary LOB files may not be deleted from the user's system.
- The enhanced LOB support applies only to non-z/OS platforms.

## Data Conversions

Teradata PT's ability to convert data is limited to assigning a data type to a null value or changing data from a null value. For example:

```
CAST (NULL AS INTEGER)
```

Using the CAST clause, you can convert to an alternate data type prior to loading data into a table. The following APPLY statement illustrates this option.

```
APPLY
('INSERT INTO CUSTOMER (:CUST_NUM, :LAST_NAME, :FIRST_NAME, :SOC_SEC_NO);')
TO OPERATOR (LOAD_OPERATOR [1] )

SELECT * FROM OPERATOR
(EXPORT_OPERATOR [1]...)
```

Here, the use of SELECT \* implies data is accepted as is from the Export operator; however, the data can also be converted.

If data is needed in a different table, create the following APPLY statement:

```

APPLY
('INSERT INTO CUSTOMER (:CUST_NUM, :LAST_NAME, :FIRST_NAME, :SOC_SEC_NO);')
TO OPERATOR (LOAD_OPERATOR [1] )
SELECT
    CAST (NULL AS CHAR(10)) AS CUST_NUM,
    LAST_NAME,
    FIRST_NAME,
    CASE
        WHEN (SOC_SEC_NO <> '000000000')
        THEN SOC_SEC_NO
        ELSE NULL
    END AS SOC_SEC_NO,
FROM OPERATOR
(EXPORT_OPERATOR [1]...

```

Notice that:

- This example assumes that the schema of the source data is not the same as the schema of the target table.
- The first field is a derived column with a NULL value.
- The target Social Security number is assigned the NULL value if the source Social Security number is a string of all zero characters.
- This use of the CASE expression is comparable to the NULLIF function of the FastLoad utility.
- The functionality provided by the CASE expression is available to all Teradata PT operators because expressions are allowed in the APPLY statement.

## Supporting User-Defined-Types and User-Defined-Methods

This section provides a brief overview of how Teradata PT supports user-defined custom data types known as User-Defined Types (UDTs), and user-defined custom functions known as User-Defined Methods (UDMs).

For specific information on using UDTs and UDMs:

- *Teradata Vantage™ - Database Introduction*, B035-1091
- *Teradata Vantage™ - SQL External Routine Programming*, B035-1147
- *Teradata Vantage™ - SQL Fundamentals*, B035-1141

## User-Defined-Types (UDTs)

UDTs can be created to provide representations of real world entities within Vantage. Choosing a set of UDTs that closely matches the entities encountered in a given business can yield a system that is easier to understand and hence easier to maintain. Support for UDTs and UDMs integrates the power of object-oriented technology directly into Vantage.

## User-Defined-Methods (UDMs)

Database developer-created custom functions, which are explicitly connected to UDTs, are known as User-Defined-Methods (UDMs). All UDMs must reside on the database.

UDMs can be used to create an interface to the UDT that is independent of the internal representation of the UDT. This makes it possible to enhance a given UDT, even in cases where the internal representation of the UDT must be changed to support the enhancement, without changing all the database applications that use the UDT.

## Creating UDTs with Teradata PT

Teradata PT cannot create a custom UDT.

## Inserting and Retrieving UDTs with Client Products

A UDT can only exist on the database. Each UDT has an associated “from-sql routine” and “to-sql routine”.

- 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 database.
- 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 database 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 database 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 database will convert the values from VARCHAR(46) to FULLNAME values using the “to-sql routine” associated with the FULLNAME UDT.

## Inserting UDTs with Teradata PT

Teradata PT inserts values into tables containing UDT columns in the same manner as it does with other tables and handles the column data in its external type format.

Data must be in the external type format in the input data source identified in the DEFINE command in the Teradata PT job script. Therefore, specify the name of the external type for the data type of the field in the DEFINE command.

## Retrieving UDTs with Teradata PT

Teradata PT can retrieve values from tables containing UDT columns in the same manner as it does with other tables.

If the select-list of the SELECT statement used in the Teradata PT job contains a UDT expression, the database automatically converts the UDT data to its external type before returning the data to Teradata PT.

As such, the data written to the output location referenced by the ".RETRIEVE" command is in the external type associated with the UDTs.

## Retrieving UDT Metadata with Teradata PT

Teradata PT cannot retrieve UDT Metadata.

## Supporting User-Defined-Functions and External-Stored-Procedures

This section provides a brief overview of how Teradata PT supports User-Defined-Functions (UDFs), and External-Stored-Procedures (XSPs).

For specific information on using UDTs and UDMs:

- *Teradata Vantage™ - Database Introduction*, B035-1091
- *Teradata Vantage™ - SQL External Routine Programming*, B035-1147
- *Teradata Vantage™ - SQL Fundamentals*, B035-1141

## User-Defined-Functions (UDFs)

Analytics Database provides a wide range of built-in functions that address many of the needs of typical users. In those cases when standard functions are inadequate, Analytics Database allows you to define user-defined functions (UDFs) that extend the capability of normal SQL within the database.

UDFs are database objects that you build or acquire. For example, you can install UDF objects or packages from third party vendors without providing the source code. A UDF is similar to standard SQL functions such as SQRT, ABS, or TRIM and is invoked in the same manner.

You can write UDFs in C or C++, then compile them into shared objects. Once compiled, you can use the UDF in SQL statements for activities such as enforcing business rules or aiding in the transformation of data.

## External-Stored-Procedures (XSPs)

Analytics Database supports external stored procedures. You can write your own external stored procedures in the C or C++ programming language, install them on the database, and then use the SQL CALL statement to call them like other stored procedures.

## Creating UDFs with Teradata PT

Teradata PT provides an option to create an UDF or an XSP in the database by using the DDL operator.

Teradata PT supports the creation of both client- and server-side UDFs/XSPs

The following apply statement creates a server side UDF named 'yourudf' by using the C file, /root/yourCudf.c residing in the /root directory of the database system:

```

APPLY
(
  CREATE FUNCTION yourudf(
    parameter_1 CHAR(1))
  RETURNS CHAR
  LANGUAGE C
  NO SQL
  EXTERNAL NAME 'SS!yourudf!/root/yourCudf.c'
  PARAMETER STYLE SQL;
)
TO OPERATOR ( $DDL );

```

If the C file exists on the client side in the current working directory, this example can be modified to support the client side UDF as follows:

```
EXTERNAL NAME ''CS!yourudf!yourCudf.c''
```

This example can also be written as:

```
EXTERNAL NAME ''CS!yourudf!yourCudf''
```

In this Case, the Teradata PT first looks for a file named 'yourCudf' in the current directory and if not found, it also looks for the filename with a .c extension 'yourCudf.c'.

**Note:**

The C, C++, or Java files can reside on the server or the client. The server calls for the transfer of client-resident files as needed for input requirements for the CREATE/REPLACE FUNCTION request. Source files must be encoded as ASCII (workstation) or EBCDIC (mainframe), regardless of the current session character set. Note, the same transfer protocol is employed when using client-resident files to create external stored procedures (XSPs).

The full specification of the syntax, format, and rules for both creating and invoking UDFs and XSPs is beyond the scope of this document. See *Teradata Vantage™ - SQL External Routine Programming*, B035-1147 for more information on how to create and invoke user-defined functions.

## Connection String

Starting in TTU 17.10, Teradata PT supports the Connection String feature. A connection string allows Teradata PT to pass various parameters and associated values to CLIV2 without the need to implement individual operator attributes for each parameter.

This enhancement allows Teradata PT to support a variety of features, one of which is the Transport Layer Security (TLS) feature.

Teradata PT operators will not parse or validate the contents of the connection string. The connection string is passed as is to CLIV2. Any errors in the connection string will be provided by CLIV2.

For a list of valid parameters and associated values, see *Teradata® Call-Level Interface Version 2 Reference for Workstation-Attached Systems*, B035-2418.

The following Teradata PT components support the Connection String feature:

- DDL operator
- Load operator
- SQL Inserter operator
- SQL Selector operator
- Stream operator
- Update operator
- EasyLoader
- Error Table Extractor

- TPTAPI
- 

**Note:**

The command is supported on all platforms, except z/OS.

---

## Identity Token Support

In z/OS 2.4, RACF introduced a new assertion mechanism, the JSON Web Token, also known as an Identity Token, which can be used to validate distributed users who have a valid RACF user id and provide a secure way to logon to the database server without specifying a password.

Starting in TTU 17.20, z/OS TPT users employing Gateway-mediated CLI can use the new RACFJWT logon mechanism that supports Identity Token.

To use the functionality in a Teradata PT job, you need to:

- Set the TPT's LogonMech option to 'RACFJWT'.
  - Set the TPT's UserName option to a valid value.
  - Do not set a value for the TPT's UserPassword option.
- 

**Note:**

TPTAPI applications must link with the "ac=1" option.

---

The functionality applies to the following TPT components:

- DDL operator
- Load operator
- Export operator
- SQL Inserter operator
- SQL Selector operator
- Stream operator
- Update operator
- Error Table Extractor
- TPTAPI

For more details on the functionality, see "Teradata Call-Level Interface Version 2 Reference for Workstation-Attached Systems".

# Extended Character Sets

## Teradata PT-Supported Character Sets

The following character sets are supported by Teradata PT:

### Workstation-Attached Character Sets

- ASCII
- UTF-8
- UTF-16
- KANJI\_EUC\_0U
- KANJI\_JIS\_0S
- KANJI932\_1S0
- SCHGB2312\_1T0
- SCHINESE936\_6R0
- TCDBIG5\_1R0
- HANGULKSC5601\_2R4
- LATIN1250\_1A0

### Mainframe-Attached Character Sets

- EBCDIC
- KATAKANA\_EBCDIC
- KANJI\_EBCDIC5026\_0I
- KANJI\_EBCDIC5035\_0I
- TCHE\_EBCDIC937\_3IB
- SCHE\_EBCDIC935\_2IJ
- HANGUL\_EBCDIC933\_1II

## Using Extended Character Sets

Teradata PT allows use of extended multibyte character sets, such as Japanese, Chinese, Korean, Latin, UTF-8, and UTF-16, as long as they are supported and defined for the database.

The character set must be common among the following elements when using Teradata PT:

- Data
- Session character set, as specified in the job script.

- Character set used to encode the job script.

The exception to the preceding rule is that for UTF-8 and UTF-16, the job script may be coded in one character set while the data is in the other, if the proper setup is executed.

For information about UTF-8 / UTF-16 compatibility and setup, see [Options for Specifying UTF-8 and UTF-16 in a Job Script](#).

For a list of valid values for session character set support, see *Teradata Vantage™ - Analytics Database International Character Set Support*, B035-1125.

---

**Note:**

UTF-8 and UTF-16 are only supported by Teradata PT on workstation-attached platforms.

---

## Character Set Hierarchy

Teradata PT follows strict hierarchies for determining which character set is used.

### Communication Sessions

- For communication sessions between Teradata PT and the database, the following rules apply in the following order:
  - **User-Determined:** The character set specified in the USING CHARACTER SET clause placed immediately before the DEFINE JOB statement is used all other character set definitions.
  - **System Parameter Block (SPB):** If no user-specified character set is specified, the character set is defined in the HSHSPB file on mainframe-attached systems or in the clispb.dat file on workstation-attached systems.
  - **Database:** If neither of the mentioned is specified, Teradata PT uses the character set defined in the database. This rule applies to the following operators that communicate with the database:
    - Load
    - Update
    - Export
    - Stream
    - SQL Inserter
    - SQL Selector
    - DDL
- **Data Transmission:** If the character set is not specified in a job that uses the DataConnector operator, Teradata PT uses the platform default character set of ASCII or EBCDIC to export data to the database.

## Specifying an Extended Character Set in a Teradata PT Job Script

**Session Character Set:** the session character set in a Teradata PT script is determined in one of the following ways:

- Default: It is not necessary to specify any session character set before defining the job if you are using the default session character set. The default session character set for all Teradata PT jobs are:
  - ASCII for workstation-attached client systems
  - EBCDIC for mainframe-attached client systems
- Specified: To use a session character set other than ASCII or EBCDIC, specify the session character set by including a session character set identification clause:

```
USING CHARACTER SET <characterSet>
```

The identification clause is specified right before the DEFINE JOB <jobname> statement.

For example:

```
DEFINE JOB LOAD_TABLES
(
  ...
  ...
  ...
);
```

In this example, the session character set will be set to the default session character set: ASCII (network) and EBCDIC (mainframe).

In the following example, the session character set, an extended session character set, will be set to KANJISJIS\_0S:

```
USING CHARACTER SET KANJISJIS_0S
DEFINE JOB LOAD_TABLES
(
  ...
  ...
  ...
);
```

**Extended Identifiers:** an identifier inside a Teradata PT script is non-keyword name that is used to uniquely identify an object. For example:

- DEFINE JOB JOB\_NAME  
where JOB\_NAME is the identifier.
- DEFINE OPERATOR FILE\_READER  
where FILE\_READER is the identifier.
- STEP Setup\_Tables  
where Setup\_Tables is the identifier.

Teradata PT allows identifiers inside a job script to contain extended characters, provided they match the extended session character set. However, when using extended identifiers in Teradata PT, delimit the extended identifier with double quotes. For example:

```
DEFINE JOB "JOB_NAME_WITH_KANJISJIS_0S_CHARACTERS"
DEFINE OPERATOR "FILE_READER_WITH_UTF8_CHARACTERS"
STEP "Setup_Tables_with_TCHBIG5_1R0_characters"
```

## Extended Session Character Set Data

Data to be loaded and exported must be the same as the session character set. The same is true for extended character sets. Data must match the extended session character set.

## Specifying Unicode in the tbuild Command

The USING CHARACTER SET <characterSet> statement in the Teradata PT job script is used to define the session character set. The session character set must match the data and it must match the encoding of the job script.

---

### Note:

USING CHARACTER SET UTF8 or UTF16 specifier is required in the script for the ADJUST UNICODE specifier to operate correctly in the DEFINE SCHEMA section, if it is absent, the ADJUST SCHEMA keyword will be ignored. For details, see [DEFINE SCHEMA](#).

When submitting a job script that is encoded in UTF-16, however, you must also specify the -e command line option for the tbuild command.

```
tbuild -f <filename> [-v jobVariableFile] -e UTF16
```

-e UTF16 indicates to Teradata PT that the job script is encoded in UTF-16. The file endianness is determined by the Byte Order Mark (BOM) at the beginning of the file.

---

### Note:

The following -e options support the different encoding schemes:

1. UTF-16 / UTF16 and any upper/lower case variations with or without a hyphen. For UTF-16 scripts: if the script is not UTF-16, error is reported. If the script endianness differs from the platform encoding, the script is converted to the platform endianness before execution.
2. UTF-16LE / UTF16LE and any upper/lower case variations with or without a hyphen. For UTF-16 little endian scripts: the script is not little endian, an error is reported. If the platform is big endian, the script is converted to big endian before execution.

3. UTF-16BE / UTF16BE and any upper/lower case variations with or without a hyphen. For UTF-16 big endian scripts: if the script is not big endian, an error is reported. If the platform is little endian, the script is converted to little endian before execution.
4. UTF-8 / UTF8 and any upper/lower case variations with or without a hyphen. For UTF-8 scripts: if the script is not UTF-8, error is reported.

The job variable and include files in either big endian or little endian format can be executed on either kind of platform.

## Setting the Byte Order Mark

Use this feature to write a BOM at the beginning of a data file. The actual BOM written is determined by the character set in use and, when using UTF-16, the endian aspect of the active platform. BOMs are detected when data is read back to ensure correct processing.

The following conditions must be met:

- The operator must be a consumer
- A Unicode character set must be specified in the script (for example, USING CHARACTER SET UTF8)
- The data format must be either text or delimited (for example, VARCHAR Format = 'Delimited')
- The WriteBOM attribute must be set to 'Yes' (for example, VARCHAR WriteBOM = 'Yes')

### Note:

If WriteBOM = 'Yes' and any of the other necessary conditions are not met, the job fails.

## Usage Notes

Consider the following when working with varying session character sets:

When using UTF-16 character set in Teradata PT scripts, the value of n in VARCHAR(n) and CHAR(n) in the SCHEMA definition must be an even and positive number.

LONG VARCHAR and LONG VARGRAPHIC are no longer supported as column types. LONG VARCHAR now corresponds to VARCHAR(64000). See [Using LONG VARCHAR with Unicode Character Sets](#) for a discussion on how Teradata PT will handle this column type.

## Options for Specifying UTF-8 and UTF-16 in a Job Script

In most cases, all the elements of a job must use a common character set:

- Data
- Session character set, as specified in the job script
- Character set used to encode the job script

However, because of the unique properties of the Unicode character sets supported by Teradata PT on workstation-attached platforms (UTF-8 and UTF-16), it is possible for the session character set and the job script encoding to use either the same or different Unicode character sets.

Specify the correct Unicode character set in a job script according to the following scenarios:

Job Script Encoding	Session Character Set	Character Set Specification
UTF8	UTF-8	Specify the character set in the job script as follows: <code>USING CHARACTER SET UTF8 DEFINE JOB...</code>
UTF8	UTF-16	Specify the character set in the job script as follows: <code>USING CHARACTER SET UTF16 DEFINE JOB...</code>
UTF16	UTF-8	Do both of the following: <ul style="list-style-type: none"> <li>Specify the character set in the job script as follows:<code>USING CHARACTER SET UTF8 DEFINE JOB...</code></li> <li>Specify the character set on the command line as follows:<code>tbuild -e UTF16</code></li> </ul>
UTF16	UTF-16	Do both of the following: <ul style="list-style-type: none"> <li>Specify the character set in the job script as follows:<code>USING CHARACTER SET UTF16 DEFINE JOB...</code></li> <li>Specify the character set on the command line as follows:<code>tbuild -e UTF16</code></li> </ul>

#### Note:

Only workstation-attached clients running Teradata PT support the use of UTF-8 and UTF-16 character sets in job scripts. Mainframe-attached clients running Teradata PT do not support these character sets.

## Using LONG VARCHAR with Unicode Character Sets

Teradata supports a column type of LONG VARCHAR. When dealing with single-byte character sets (both client session character set and server storage character set) a LONG VARCHAR is interpreted as VARCHAR(64000).

When processing the script and coming across a column of type LONG VARCHAR, Teradata PT interprets this column type as VARCHAR(64000). Since the column type is passed on to the operators, some jobs may not run properly.

Problems may arise when the server side storage character set is Unicode or when the LONG VARCHAR column type is used in a schema definition. This is because the combination of the client side session character set and the server storage character set can cause the LONG VARCHAR specification in a DML USING clause to mean something other than VARCHAR(64000).

In summary:

- The use of LONG VARCHAR in a schema object definition is not recommended.
- Do not use LONG VARCHAR with Unicode character sets. Instead, specify VARCHAR(64000).

## UTF8/UTF16 Considerations

Character-type columns (CHAR, VARCHAR, CLOB, JSON) in database tables are defined in terms of characters. Client applications, such as Teradata Parallel Transporter, process data for those columns in terms of bytes.

Careful consideration should be taken when attempting to load data into character-type columns when the sizes of those columns approach more than half of the maximum allowed by the database.

For columns defined in the database as CHAR(32000) CHARACTER SET UNICODE and/or VARCHAR(32000) CHARACTER SET UNICODE:

- For Client Session Character Sets of UTF8:
  - The maximum size in a TPT script schema is 32000.
- For Client Session Character Sets of UTF16:
  - The maximum size in a TPT script schema is 64000.

For columns defined in the database as CLOB(1048544000) CHARACTER SET UNICODE and/or JSON(1048544000) CHARACTER SET UNICODE:

- For Client Session Character Sets of UTF8:
  - The maximum size in a TPT script schema is 1048544000.
- For Client Session Character Sets of UTF16:
  - The maximum size in a TPT script schema is 2097088000.

For the Stream Operator, though, since DML statements are executed through macros, there are additional limitations imposed by the Teradata Database, as follows:

For columns defined in the database as CHAR(32000) CHARACTER SET UNICODE and/or VARCHAR(32000) CHARACTER SET UNICODE:

- For Client Session Character Sets of UTF8 and UTF16:
  - The maximum size in a TPT script schema is 32000.

For columns defined in the database as CLOB(1048544000) CHARACTER SET UNICODE and/or JSON(1048544000) CHARACTER SET UNICODE:

- For Client Session Character Sets of UTF8 and UTF16:
  - The maximum size in a TPT script schema is 1048544000.

# How to Read Syntax Diagrams

## Overview

This section describes how to read syntax diagrams and syntax diagram conventions.

## Syntax Diagram Conventions

### Notation Conventions

Item	Definition and 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	Keywords and variables. <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 Linux command.</li> <li>Mixed Case letters represent exceptions to uppercase and lowercase rules. The exceptions are noted in the syntax explanation.</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 an excerpt from the diagram. The excerpt is defined immediately following the diagram that contains it.</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.

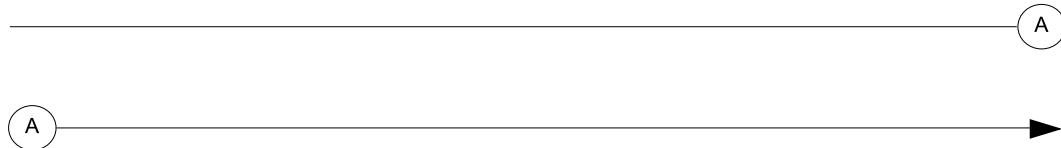
### Paths

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

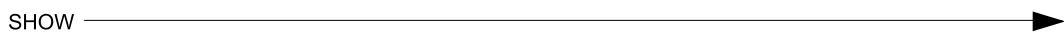
Paths that are too long for one line use continuation links. Continuation links are circled letters indicating the beginning and end of a link:



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

## Required Entries

Required entries appear on the main path:

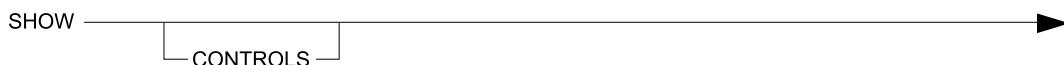


If you can choose from more than one entry, the choices appear vertically, in a stack. The first entry appears on the main path:



## Optional Entries

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



If you can optionally choose from more than one entry, all the choices appear after 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

String literals appear in apostrophes:

```
_____
| 'msgtext' |
_____
```

## Abbreviations

If a keyword or a reserved word has a valid abbreviation, the unabbreviated form always appears on the main path. The shortest valid abbreviation appears beneath.

```
SHOW _____ CONTROLS _____ →
      |           |
      |           CONTROL
```

In this 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 before the main path, over the item or items that you can repeat:



Read loops from right to left.

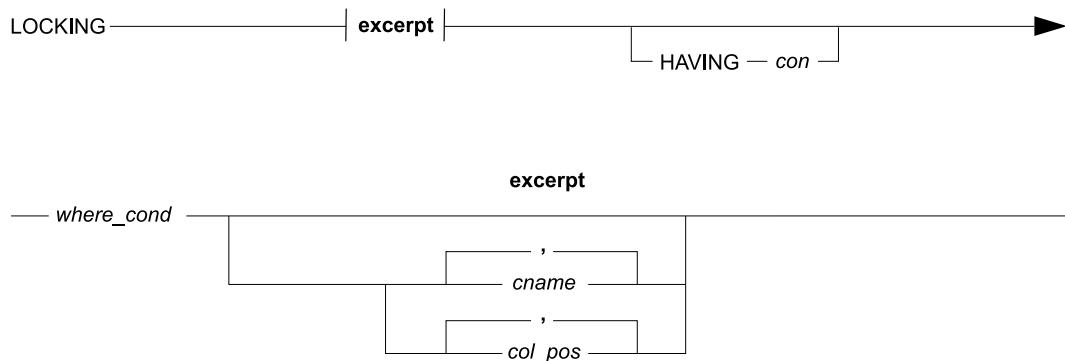
The following conventions apply to loops:

Item	Description	Example
maximum number of entries allowed	The number appears in a square on the return path.	In the example, you may type <i>cname</i> a maximum of four times.
minimum number of entries allowed	The number appears in a circle on the return path.	In the example, you must type at least three groups of column names.
separator character 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.
delimiter character 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

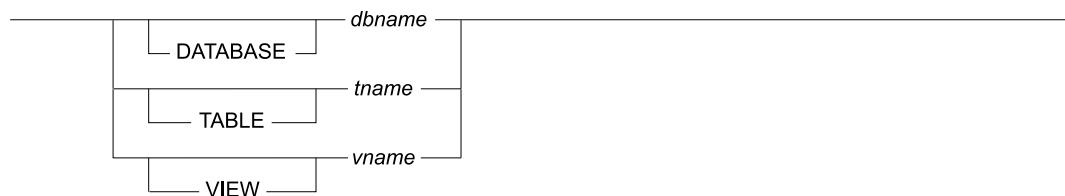
Sometimes a piece of a syntax phrase is too large to fit into the diagram. Such a phrase is indicated by a break in the path, marked by (|) terminators on each side of the break. The name for the excerpted piece appears between the terminators in boldface type.

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



## Multiple Legitimate Phrases

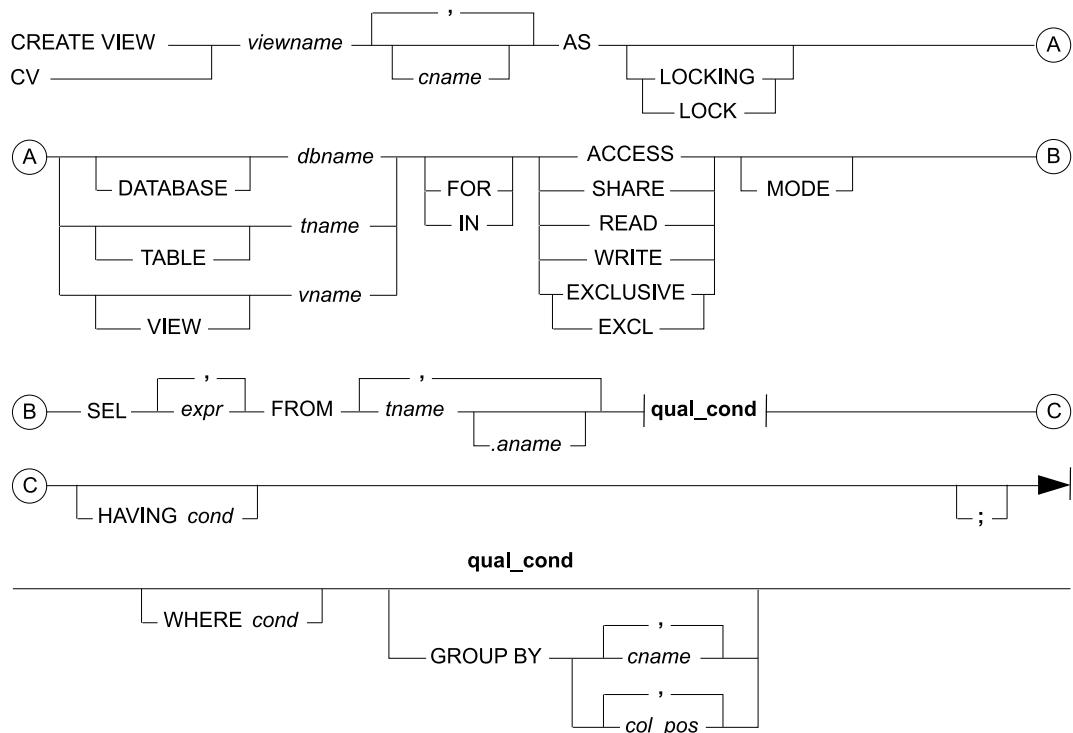
In a syntax diagram, it is possible for any number of phrases to be legitimate:



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

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

## Sample Syntax Diagram



# Deprecated Syntax

## Syntax Changes

The following syntax changes affect Teradata PT. Use the following table to understand what syntax was added, removed, and modified since Teradata WB 4.1. Deprecated syntax is still supported in the code to ensure backwards compatibility; however, it will not be documented beyond a certain release.

Syntax	Type of Statement	Status as of This Release
( )	OPERATOR	No longer required for OPERATOR statements, but if used, still functions as in Teradata WB 4.1. For more information, see the documentation for that release.
ALLOW LATENCY CHECK	OPERATOR	Obsolete syntax; do not use in OPERATOR statements.
ALLOW PARALLEL MAX INSTANCES	OPERATOR	Obsolete syntax; do not use in OPERATOR statements.
AmpCheck	TDLOAD LONG_OPTION	Obsolete. tdload now follows a standard syntax based on current job variable templates.
AppendErrorTable	TDLOAD LONG_OPTION	Obsolete. tdload now follows a standard syntax based on current job variable templates.
APPLY	OPERATOR	New, required keyword that replaces previous LOAD syntax in OPERATOR statements. For more information, see <a href="#">Object Definitions and the APPLY Statement</a> .
ArraySupport	TDLOAD LONG_OPTION	Obsolete. tdload now follows a standard syntax based on current job variable templates.
ATTRIBUTE   ATTR (attributeDefinition List)	JOB	No longer required for OPERATOR statements, but if used, still functions as in Teradata WB 4.1. For more information, see the documentation for that release.
BlockSize	TDLOAD LONG_OPTION	Obsolete. tdload now follows a standard syntax based on current job variable templates.

Syntax	Type of Statement	Status as of This Release
Buffer	TDLOAD LONG_OPTION	Obsolete. <code>tdload</code> now follows a standard syntax based on current job variable templates.
CONSUMER		Do not use. See “Type Definitions” in the <i>Teradata® Parallel Transporter User Guide</i> , B035-2445 .
Column Definition List	OPERATOR TABLE TABLE SET	Obsolete syntax; do not use in OPERATOR, TABLE, or TABLE SET statements.
DEFINE DBMS	OPERATOR	Obsolete syntax; do not use.
DEFINE FILE ' <i>pathName</i> '	JOB	Obsolete syntax; do not use in JOB statements.
DEFINE LIBRARY PATH ' <i>pathName</i> '	JOB	Obsolete syntax; do not use in JOB statements.
DEFINE LOGGER	LOGGER	Obsolete syntax; do not use. Instead, use <code>tlogview</code> .
DEFINE LOG VIEW	LOG VIEW	Obsolete syntax; do not use. Instead, use <code>tlogview</code> .
DEFINE SCHEMA <i>schema name</i> <i>'DBS table name'</i>	SCHEMA	Obsolete syntax; do not use.
DEFINE SCHEMA <i>schema name</i> DELIMITED ' <i>DBS table name</i> '	SCHEMA	Obsolete syntax; do not use.
DEFINE SYSTEM PATH ' <i>path name</i> '	JOB	Obsolete syntax; do not use in JOB statements.
DeleteTask	TDLOAD LONG_OPTION	Obsolete. <code>tdload</code> now follows a standard syntax based on current job variable templates.
DISALLOW LATENCY CHECK	OPERATOR	Obsolete syntax; do not use in OPERATOR statements.
DropErrorTable	TDLOAD LONG_OPTION	Obsolete. <code>tdload</code> now follows a standard syntax based on current job variable templates.
DropLogTable	TDLOAD LONG_OPTION	Obsolete. <code>tdload</code> now follows a standard syntax based on current job variable templates.
DropWorkTable	TDLOAD LONG_OPTION	Obsolete. <code>tdload</code> now follows a standard syntax based on current job variable templates.

Syntax	Type of Statement	Status as of This Release
ErrorTable	TDLOAD LONG_OPTION	Obsolete. <code>tdload</code> now follows a standard syntax based on current job variable templates.
ErrorTable1	TDLOAD LONG_OPTION	Obsolete. <code>tdload</code> now follows a standard syntax based on current job variable templates.
ErrorTable2	TDLOAD LONG_OPTION	Obsolete. <code>tdload</code> now follows a standard syntax based on current job variable templates.
ExportPrivateLogName	TDLOAD LONG_OPTION	Obsolete. <code>tdload</code> now follows a standard syntax based on current job variable templates.
EXTERNAL NAME ' <i>operatorFileName</i> '	OPERATOR	No longer required when using predefined operators (for example, TYPE LOAD, TYPE EXPORT, TYPE STREAM). Required only when using TYPE CONSUMER, TYPE PRODUCER, TYPE FILTER, and TYPE STANDALONE syntax.
FILE ' <i>fileName</i> '	TABLE TABLE SET	Obsolete syntax; do not use in TABLE or TABLE SET statements.
FILTER		Do not use. See “Type Definitions” in the <i>Teradata® Parallel Transporter User Guide</i> , B035-2445.
GENERIC	DBMS	Obsolete syntax; do not use in DBMS statements.
IgnoreMaxDecimalDigits	TDLOAD LONG_OPTION	Obsolete. <code>tdload</code> now follows a standard syntax based on current job variable templates.
INPUT/OUTPUT SCHEMA	OPERATOR	The “INPUT” and “OUTPUT” portion of this syntax is ignored by the compiler unless a filter operator contains both INPUT SCHEMA and OUTPUT SCHEMA. In this case, use “INPUT” and “OUTPUT” before “SCHEMA” to differentiate between the two schemas.
INSERT OPERATOR ' <i>operatorName</i> '	DBMS TABLE SET	Obsolete syntax; do not use in DBMS or TABLE SET statements. If present in an existing script, this syntax is treated as the APPLY OPERATOR ' <i>operatorName</i> ' syntax. Instead, use APPLY OPERATOR ' <i>operatorName</i> '. For more information, see <a href="#">Object Definitions and the APPLY Statement</a> .
INTERFACE PXOPER VERSION ' <i>versionIdentifier</i> '	OPERATOR	Obsolete syntax; do not use in OPERATOR statements.

Syntax	Type of Statement	Status as of This Release
LANGUAGE C   C++	OPERATOR	Obsolete syntax; do not use in OPERATOR statements.
LIBRARY PATH ' <i>pathName</i> '	OPERATOR	Obsolete syntax; do not use in OPERATOR statements.
LOAD	JOB	Obsolete syntax; do not use in JOB statements. Use of this keyword now results in a compiler error. Instead, use the APPLY statement. For more information, see <a href="#">Object Definitions and the APPLY Statement</a> .
LogTable	TDLOAD LONG_OPTION	Obsolete. tdload now follows a standard syntax based on current job variable templates.
LONGVARCHAR		<p>Keyword specifying a fixed-length character string data type of up to 64,000 bytes.</p> <p>Using this keyword in Teradata PT creates a conflict between the line size limits of the designated input character length and what is output because Teradata PT recognizes double-byte characters while Vantage does not.</p> <p>Because of this conflict, avoid using this keyword, and use VARCHAR(64000) or VARCHAR(32000) instead.</p>
LONGVARCHARGRAPHIC		<p>Keyword specifying a fixed-length character string data type of up to 32,000 bytes.</p> <p>Using this keyword in Teradata PT creates a conflict between the line size limits of the designated input character length and what is output because Teradata PT recognizes double-byte characters while Vantage does not.</p> <p>Because of this conflict, avoid using this keyword, and use VARCHAR(64000) or VARCHAR(32000) instead.</p>
MacroDatabase	TDLOAD LONG_OPTION	Obsolete. tdload now follows a standard syntax based on current job variable templates.
MaxDecimalDigits	TDLOAD LONG_OPTION	Obsolete. tdload now follows a standard syntax based on current job variable templates.
MSGCATALOG ' <i>catalogName</i> '	OPERATOR	<p>Obsolete syntax; do not use in OPERATOR statements when specifying a new predefined operator syntax (for example, TYPE LOAD, TYPE EXPORT, TYPE STREAM).</p> <p>Use only when specifying a generic operator type (such as TYPE CONSUMER) if the name of the default message catalog changed.</p>

Syntax	Type of Statement	Status as of This Release
		When used, still functions as in Teradata WB 4.1. For more information, see the documentation for that release.
MULITPHASE	OPERATOR	Obsolete syntax; do not use in OPERATOR statements.
N'...'	All character string literals	Obsolete syntax; do not use the initial "N" to specify an extended character inside a character string literal.
N"..."	All character identifiers	Obsolete syntax; do not use the initial "N" to specify an extended character inside a character string literal.
PATH ' <i>pathName</i> '	DBMS TABLE SET	No longer required for DBMS or TABLE SET statements, but if used, still functions as in Teradata WB 4.1. For more information, see the documentation for that release.
OpenMode	TDLOAD LONG_OPTION	Obsolete. tdload now follows a standard syntax based on current job variable templates.
Pack	TDLOAD LONG_OPTION	Obsolete. tdload now follows a standard syntax based on current job variable templates.
PackMaximum	TDLOAD LONG_OPTION	Obsolete. tdload now follows a standard syntax based on current job variable templates.
PauseAcq	TDLOAD LONG_OPTION	Obsolete. tdload now follows a standard syntax based on current job variable templates.
Periodicity	TDLOAD LONG_OPTION	Obsolete. tdload now follows a standard syntax based on current job variable templates.
PRODUCER		Do not use. See "Type Definitions" in the <i>Teradata® Parallel Transporter User Guide</i> , B035-2445.
QueueErrorTable	TDLOAD LONG_OPTION	Obsolete. tdload now follows a standard syntax based on current job variable templates.
Rate	TDLOAD LONG_OPTION	Obsolete. tdload now follows a standard syntax based on current job variable templates.
ReplicationOverride	TDLOAD LONG_OPTION	Obsolete. tdload now follows a standard syntax based on current job variable templates.

Syntax	Type of Statement	Status as of This Release
Robust	TDLOAD LONG_OPTION	Obsolete. <code>tdload</code> now follows a standard syntax based on current job variable templates.
RESTART /NOT RESTARTABLE	OPERATOR	No longer required for OPERATOR statements, but if used, still functions as in Teradata WB 4.1. For more information, see the documentation for that release.
RowErrFileName	Attribute	<code>RecordErrorFilename</code> is the new attribute name. For this release, both attribute names are mentioned; however, in future releases, only <code>RecordErrorFilename</code> will be mentioned in the documentation.
SelectStmt	TDLOAD LONG_OPTION	Obsolete. <code>tdload</code> now follows a standard syntax based on current job variable templates.
SET CHECKPOINT INTERVAL	JOB	Use the <code>-z</code> option with the <b>tbuild</b> command. See the <i>Teradata® Parallel Transporter User Guide</i> , B035-2445.
SET LATENCY INTERVAL	JOB	Use the <code>-l</code> option with the <b>tbuild</b> command. See the <i>Teradata® Parallel Transporter User Guide</i> , B035-2445.
STANDALONE		Do not use. See “Type Definitions” in the <i>Teradata® Parallel Transporter User Guide</i> , B035-2445.
TenacityHours	TDLOAD LONG_OPTION	Obsolete. <code>tdload</code> now follows a standard syntax based on current job variable templates.
TenacitySleep	TDLOAD LONG_OPTION	Obsolete. <code>tdload</code> now follows a standard syntax based on current job variable templates.
WorkTable	TDLOAD LONG_OPTION	Obsolete. <code>tdload</code> now follows a standard syntax based on current job variable templates.

# Restricted Words

## Overview

Restricted words have special meaning for use in Teradata PT. They must not be used in Teradata PT job scripts as identifiers for column names, attributes, or other values.

Following is a list of the Teradata PT restricted words. The words are shown in uppercase type for simplicity, but are restricted regardless of case.

---

### Note:

As of this release, Teradata PT reserved keywords are now called Teradata PT restricted words.

---

For a full list of Teradata reserved, non reserved, and future words, see *Teradata Vantage™ - SQL Fundamentals*, B035-1141.

ACCOUNT	ELSE	MACROCHARSET	SCHEMAMAPPER
ALL	ENABLE	MARK	SEC
ALLOW	END	MAX	SECOND
AND	EXECUTE	MAXIMUM	SECONDS
ANSIDATE	EXPORT	MESSAGE	SELECT
APPLY	EXTERNAL	METADATA	SELECTOR
ARRAY	EXTRA	MIN	SEQUENTIAL
AS	FALSE	MINUTE	SERIALIZE
ATTR	FASTEXPORT	MINUTES	SET
ATTRIBUTES	FASTLOAD	MISSING	SMALLINT
BIGINT	FILE	MODE	SOURCE
BLOB	FILTER	MONTH	STANDALONE
BY	FLOAT	MSGCATALOG	STEP
BYTE	FOR	MSGNUMBER	STREAM
BYTEINT	FORMATIN	MSGTEXT	SUPPORT
CASE	FORMATOUT	MULTILOAD	SYSTEM
CAST	FROM	MULTIPHASE	TABLE
CATEGORY	GENERIC	NAME	TASKID
CHAR	GRAPHIC	NODE	TASKNAME
CHARACTER	HOUR	NODEBUG	TERADATA
CHARACTERS	ID	NODENAME	THEN
CHARS	IF	NOINFO	TIME
CHECK	IGNORE	NOT	TIMESTAMP
CHECKPOINT	INCLUDE	NOTRACE	TO
CLOB	INFO	NULL	TRACE
COMMAND	INMOD	NULLIF	TRUE
COMPONENT	INPUT	NUMBER	TYPE

CONSUMER	INSERT	NUMERIC	UNION
DATABASE	INSERTER	ODBC	UNKNOWN
DATACONNECTOR	INSTANCE	OFF	UPDATE
DATAGENERATOR	INT	OFFSET	USE
DATE	INTDATE	ON	USER
DATETIME	INTEGER	OPERATOR	USING
DAY	INTERFACE	OR	VARBYTE
DBMS	INTERVAL	OS	VARCHAR
DDL	INTO	OTHER	VARDATE
DEBUG	IS	OUTMOD	VARGRAPHIC
DEC	JOB	OUTPUT	VARYING
DECIMAL	JSON	PARALLEL	VERSION
DEFAULT	KEEP	PASSWORD	VIA
DEFERRED	LANGUAGE	PATH	VIEW
DEFINE	LATENCY	PERIOD	WHEN
DELETE	LENGTH	PROCESSID	WHERE
DELIM	LIBRARY	PRODUCER	WITH
DELIMITER	LOAD	PXOPER	WORKING
DELTATIME	LOCAL	REMOTE	XML
DESCRIPTION	LOG	RESTARTABLE	XSMOD
DIRECTORY	LOGGER	ROWS	YEAR
DISABLE	LOGON	SCHEMA	ZONE
DISALLOW	LOGREPORT		
DISCARD	LONG		
DUPLICATE			

# Teradata PT Publications

## Teradata PT Reference Documentation

User documentation for Teradata Parallel Transporter is distributed among the following documents:

Publication	Contents
<i>Teradata® Parallel Transporter Application Programming Interface Programmer Guide</i> , B035-2516	Provides information about: <ul style="list-style-type: none"> <li>Setting up the interface.</li> <li>Coding.</li> <li>Error reporting.</li> <li>Checkpointing and restarting.</li> </ul>
<i>Teradata® Parallel Transporter Operator Programmer Guide</i> , B035-2435	Provides information on developing custom operators, including all interface functions that allow communication between the Teradata PT operators and the Teradata PT infrastructure.
<i>Teradata® Parallel Transporter Quick Start Guide</i> , B035-2501	Provides getting-started information for using Teradata PT. Includes Teradata PT job examples for: <ul style="list-style-type: none"> <li>Reading data from a flat file and loading it into a database target table.</li> <li>Exporting data from a database source table and writing it to a flat file.</li> <li>Exporting data from a database source table and loading it to a database target table.</li> </ul>
<i>Teradata® Parallel Transporter Reference</i> , B035-2436 (this document)	A reference document that defines: <ul style="list-style-type: none"> <li>Teradata PT command line utility commands.</li> <li>Object definition statements that make up the declarative section of a Teradata PT job script.</li> <li>The APPLY statement that makes up the executable section of a Teradata PT job script.</li> <li>Syntax for each Teradata PT operator.</li> </ul>
<i>Teradata® Parallel Transporter User Guide</i> , B035-2445	Detailed strategies for planning, implementing, and debugging Teradata PT. The document includes topics on: <ul style="list-style-type: none"> <li>Writing Teradata PT template job scripts, the kind of job scripts illustrated in the <i>Teradata® Parallel Transporter Quick Start Guide</i></li> <li>Writing Teradata PT defined schema job scripts that:               <ul style="list-style-type: none"> <li>Move data to and from data targets</li> <li>Move data within the database environment</li> </ul> </li> <li>Describing individual Teradata PT operators and access modules</li> <li>Launching, managing, and troubleshooting a Teradata PT job</li> </ul>

# Stream Operator Performance Tuning

## Overview

There are three categories of variables that can and will impact the performance of a given TPT job using the Stream operator:

- Variables defined within the TPT script using the Stream operator itself
- Data-related variables
- and environmental variables

## Attributes in the TPT Script That Use the Stream Operator

For the most part, you may choose how to set the attributes in the Stream operator itself. The key variables are Pack attribute, MaxSessions attribute, MinSessions attribute, ArraySupport attribute (ON or OFF) and SERIALIZED job option (ON or OFF).

- The **Pack** attribute – is one of the key performance enablers in the Stream operator, represents the number of rows that each session's buffer will hold. Its default value is 20. In general, a higher pack factor usually yields better performance than a smaller pack factor does.
- The **MaxSessions** and **MinSessions** attributes – tell the Stream operator the maximum and minimum number of sessions to transfer data to the database.
- The **ArraySupport** attribute – offers a new data-driven iteration capability that provides an improved way for the Stream operator to iterate a parameterized DML statement for multiple sets of parameter values within a single request.
- **SERIALIZED** job option – can be used to ensure the order of application of records and/or to reduce row hash lock contention. The Stream operator will calculate a hash value based on the key(usually the primary index) to determine the session assigned to process each input row. This allows all rows with the same key to be processed in sequence by the same session, which is especially important if rows are distributed among many sessions.

These variables are correlated. Here are some recommendations:

- A general Stream operator throughput recommendation is “Pack up to maximum and Sessions Up until trouble”. If experiencing AWT, CPU, or hash collision problems, reduce sessions; use SERIALIZED ON, so session reduction should not be too far beyond 8 or 16 from current/recommended session counts.
- Keep MaxSessions/MinSessions the same to help serialization. Starve sessions with lower Pack factor to slow the row rates. Reduce packs by 15% for tables with medium-to-large row size; reduce packs by 25% for tables with small row size.
- Records with same primary index hash to the same session. Data “clumps” with fewer sessions can slow throughput
- Increasing Pack with the same session footprint will increase rows-per-second throughput.

- With SERIALIZED ON, session count needs to be kept as an odd number.

**Note:**

When the following conditions are met:

- The schema has variable-length columns and
- Array Support is on and
- One of the following:
  - The PACK factor is set to 2400
  - The PACKMAXIMUM is set to 'Y[es]'
  - Neither the PACK factor attribute nor PACKMAXIMUM attribute is populated

The user will be informed the "floating" Pack factor via a new informational message similar to the following:

```
**** 14:50:34 The PACK factor has changed. The minimum PACK factor is about
1270 data records per request. The maximum PACK factor is about 1298 data
records per request.
```

## Data-Related Variables

Variables related to the database design and the state of the data itself include:

- How clean the data is
- Any sequence to the input stream
- Degree and type of indexes defined on the table being updated
- Use of Referential Integrity, fallback or permanent journaling
- Number and complexity of triggers
- Number of columns being passed into the database per update
- Ratio of UPDATEs to INSERTs when an UPSERT is used

The data related variables are important to performance because they may dictate how large the pack factor can be, and because these variables will influence how much effort it will take to process each row. The more secondary indexes, triggers, or other options, such as fallback or referential integrity (RI) that are defined on the table being updated, the more complex the work in the database will be to perform that update. With this additional complexity comes an increase in the database steps that are generated internally to perform the work. Because there is an internal limit on the total size of the steps generated for a single request, the Stream operator in some cases may have to reduce the pack factor when the table has been defined using some number of these table options.

The number of columns being passed into the database can also impact the Pack factor. The Stream operator uses the Teradata multi-statement request convention, in which multiple SQL statements are bundled together into a single optimizing and recovery unit. This is the most efficient approach when combined with a USING clause, so that the data goes into the database in one parcel and the SQL in a

second parcel. When the USING clause is utilized as it is with the Stream operator, the query plans are cached and throughput can be increased by bypassing the parsing activity. When UPSERT processing is expected, the ratio of UPDATEs to INSERTs impacts the Stream operator's performance. The higher the percentage of INSERTs, the lower the throughput. This is because the UPDATE has to enter the database and attempt to find the row, and fail, before the Stream operator knows to perform the INSERT instead.

## Environment Variables

Most environmental variables are difficult to control or modify. They include such things as:

- Whether the Stream operator is running on a TPA-node, non-TPA node, or an external client
- The power of the client platform
- The size of the pipe (network or channel) between the client and the database
- Number of nodes and CPU power of the server
- The Stream operator client software version
- Vantage software version

Environmental variables become particularly important to consider when you are testing in one environment, and planning on moving a Stream operator job into production in an entirely different environment. Performance expectations based on a set of the Stream operator's parameters that were optimized on a 1-node Teradata test system, are unlikely to be entirely effective when moved to a 20-node production system with a more powerful client and wider network bandwidth. Some parameter settings will have to be rethought moving from test to production.

## Tips for the Stream Operator

1. High pack factors can increase the Stream operator's throughput. When high pack factors cannot be used, increasing the number of sessions is another way to boost the Stream operator's throughput if the client can support them.
2. To reduce data load latency and improve real-time availability of single rows, reduce the pack factor.
3. If input data contains errors, a low pack factor will reduce the overhead of rolling back the request that contains the error, and re-process all error-free rows.
4. Speed up the Stream operator's startup by using persistent macros and specifying the Stream operator's recommended Pack factor from a previous/similar run.
5. When selecting the number of sessions, consider the total system load at the time the TPT job using the Stream operator is run. When multiple TPT jobs using the Stream operator are running, consider a number of sessions equal to the number of AMPs in the system, or less.
6. If multiple TPT jobs using the Stream operator may update rows from the same table with the same primary index value, manually partition the data on the primary index of the table, so all rows with the same PI value are directed to the same job. Then also specify SERIALIZED ON to force the rows with the same NUPI value to a single session within that job, further reducing possible contention.
7. If a TPT job using the Stream operator is doing INSERT operations against a target table with a join index defined, consider directing the TPT job to insert into a non-indexed staging table. An insert/select from

this staging table into the base table at regular intervals is likely to be a better-performing approach to updating a table when a join index is involved. Prior to the insert/select, a UNION can be used to make sure data recently inserted into the staging table is included in query answer sets.

8. Assign the TPT user to a higher priority performance group when the TPT job using the Stream operator runs at the same time as decision support queries, if the TPT completion time is more critical than the other work active in the system.
9. To ensure that the Stream operator is able to perform single-AMP operations on each input record, include the entire primary index value for the row being updated among the columns passed to the database.
10. Use the latest versions of client TPT and Teradata CLIV2 and the latest version of the database.

# Teradata PT Variants on z/OS

## Overview

On z/OS, TPT has two variants:

- TDP (Teradata Director Program)
- Gateway (also known as Mainframe Cloud-Network Connectivity)

## Differences between Teradata Director Program and Gateway Variants

The TDP variant of TPT uses the channel-attached CLI to interface with the Analytics Database, which requires a channel connection.

The Gateway variant of TPT uses the network-attached CLIV2 to interface with the Analytics Database, which uses the Database Gateway software.

The following table shows differences between the two variants:

For	TDP Variant	Gateway Variant
Access to Analytics Database on a cloud	No with direct channel connection	Yes
Channel connection	Yes, required	No
CLIV2	Uses channel-attached CLI	Uses network-attached CLIV2
CLIV2 error codes	Uses channel-attached CLI error codes	Uses network-attached CLIV2 error codes
Connection string	Not supported	Supported
Interface with Gateway software	No	Yes
Longer session connection time	No	Possibly
<hlq>.ETC.IPNODES member file	Not needed	Yes, required
Display of variant in a TPT log	CLIV2 TDP	CLIV2 GTW
Supported SQL Engine versions	All supported versions	17.10.00.00 or later 17.05.05.03 or later 16.20.53.31 or later

The Gateway variant of TPT can take a longer time to connect sessions, because there is more overhead between the network-attached CLlv2 and the Database Gateway software for authentication and encryption. This overhead does not exist in the TDP variant of TPT.

The Gateway variant of TPT requires COP entries in the <hlq>.ETC.IPNODES member file. The entries must define the IP address and COP alias name for each database node. <hlq> is the site defined high level qualifier data set name.

The TDP variant does not use the <hlq>.ETC.IPNODES member file.

Using the Gateway variant of TPT against un-supported Analytics Database versions might cause unexpected results.

Teradata recommends upgrading to a supported Analytics Database version.

# Additional Information

## Purpose

This document provides reference information about the components of Teradata® Parallel Transporter (Teradata PT), a Teradata Tools and Utilities product. Teradata Tools and Utilities is a group of client products designed to work with Analytics Database.

Teradata PT provides high-performance data extraction, loading, and updating operations for Vantage.

## Audience

This document is intended for use by:

- System and application programmers
- System administrators
- Database administrators
- Database developers
- System operators
- Other database specialists using Teradata PT

## Release Compatibility

This document applies to the following releases:

- Teradata Vantage™ 2.0
- Teradata Tools and Utilities Release 17.20
- Teradata Parallel Transporter 17.20

The most current information about supported operating systems, supported Vantage versions, and product version numbers for all Teradata Tool and Utilities is available at <https://docs.teradata.com/> in a single spreadsheet titled *Teradata® Tools and Utilities Supported Platforms and Product Versions* (BCD0-2815).

## Changes and Additions

Date	Release	Description
August 2022	17.20-LA	<p>In z/OS 2.4, RACF introduced a new assertion mechanism, the JSON Web Token, also known as an Identity Token. This mechanism can be used to validate distributed users who have a valid RACF user id and provide a secure way to logon to Teradata database server without specifying a password.</p> <p>On z/OS platform, TPT is available in the following two variants:</p>

Date	Release	Description
		<ul style="list-style-type: none"> <li>• Teradata Director Program (TDP)</li> <li>• Mainframe Cloud-Network Connectivity (Gateway)</li> </ul>

## Teradata Links

Link	Description
<a href="https://docs.teradata.com/">https://docs.teradata.com/</a>	Search Teradata Documentation, customize content to your needs, and download PDFs. Customers: Log in to access Orange Books.
<a href="https://support.teradata.com">https://support.teradata.com</a>	Helpful resources in one place: <ul style="list-style-type: none"> <li>• Support requests</li> <li>• Account management and software downloads</li> <li>• Knowledge base, community, and support policies</li> <li>• Product documentation</li> <li>• Learning resources, including Teradata University</li> </ul>
<a href="https://www.teradata.com/University/Overview">https://www.teradata.com/University/Overview</a>	Teradata education network
<a href="https://support.teradata.com/community">https://support.teradata.com/community</a>	Link to Teradata community

## Related Documentation

Title	Publication ID
<i>Teradata® Tools and Utilities Access Module Programmer Guide</i>	B035-2424
<i>Teradata® Tools and Utilities Access Module Reference</i>	B035-2425
<i>Teradata® Parallel Transporter Application Programming Interface Programmer Guide</i>	B035-2516
<i>Teradata® Parallel Transporter Operator Programmer Guide</i>	B035-2435
<i>Teradata® Parallel Transporter Quick Start Guide</i>	B035-2501
<i>Teradata® Tools and Utilities for IBM z/OS Installation Guide</i>	B035-3128
<i>Teradata® Tools and Utilities for Microsoft Windows Installation Guide</i>	B035-2407
<i>Teradata® Tools and Utilities for Linux Installation Guide (Amazon Linux 2, CentOS, OEL, RedHat, SLES, Ubuntu)</i>	B035-3160
<i>Teradata® Tools and Utilities for IBM AIX Installation Guide</i>	B035-3125

Title	Publication ID
<i>Teradata® Tools and Utilities for Oracle Solaris on SPARC and AMD Opteron Systems Installation Guide</i>	B035-3136