IBM InfoSphere DataStage and QualityStage
Version 9 Release 1

*Connectivity Guide for Teradata Databases*

IBM
Note

Before using this information and the product that it supports, read the information in “Notices and trademarks” on page 163.
# Contents

## Chapter 1. Connector Migration Tool
- Migrating jobs to use connectors ........ 1
  - Using the user interface to migrate jobs .... 2
  - Using the command line to migrate jobs .... 3
- Deprecated stages .......................... 6

## Chapter 2. Configuring access to Teradata databases
- Configuring access to Teradata databases .... 9
  - Testing database connections by using the ISA Lite tool .......... 9
- Setting the library path environment variable 10
  - Setting the library path environment variable in the dsenv file .. 10
  - Setting the library path environment variable in Windows .......... 11

## Chapter 3. Teradata connector
- Performance .................................. 13
- SQL statements ............................... 13
- Designing jobs that use the Teradata connector 14
  - Operators in the Teradata Parallel Transporter .. 14
  - Configuring the connector as a source ........ 16
  - Configuring the connector as a target ......... 16
  - Configuring the Logon mechanism ............ 18
  - Configuring the connector to perform lookups ... 19
- Generating SQL statements in the connector at design time .......... 20
- Validating SQL statements in the connector at design time ........ 20
- Data types .................................. 21
- Parallel synchronization table .................. 22
- Teradata Multi-System Manager Integration and Dual Load ........ 25
  - Installation and configuration requirements for the TMSM integration ... 25
  - Sample job with TMSM integration in the Teradata connector .. 26
  - Teradata connector support for TMSM .......... 27
- Options for client character set in server stages .............. 28
- Teradata connector properties .................. 29
  - Access method .......................... 29
  - Account ................................ 30
  - After SQL ................................ 30
  - After SQL file .......................... 30
  - After SQL (node) ......................... 31
  - Allow duplicate rows ..................... 32
  - Array size ................................ 32
  - Automap character encoding .............. 32
  - Before/After SQL ........................ 33
  - Before SQL ................................ 33
  - Before SQL file .......................... 33
  - Before SQL (node) ......................... 34
  - Buffer usage ............................ 35
  - Bulk access .............................. 35
  - Case sensitive ........................... 35
  - Character set ............................ 35
  - Checkpoint timeout ....................... 35
  - Cleanup mode ............................ 36
  - Client character set ...................... 36
  - Columns .................................. 37
  - Column delimiter ......................... 37
  - Create statement ......................... 38
  - Create table options ...................... 38
  - Data block size ......................... 38
  - Database ............................... 38
  - Delete multiple rows ..................... 38
  - Delete statement ......................... 39
  - Drop statement ........................... 39
  - Duplicate insert rows .................... 39
  - Duplicate update rows .................... 40
  - Enable LOB references ..................... 40
  - Enable quoted identifiers ................ 40
  - End of data ................................ 41
  - End of wave ............................. 41
  - End row .................................. 41
  - End timeout .............................. 41
  - Error control ............................ 41
  - Error limit .............................. 41
  - Error table 1 ............................ 42
  - Error table 2 ............................ 42
  - Fail on error ............................. 42
  - Fail on size mismatch ..................... 42
  - File ..................................... 43
  - Free space percent ....................... 43
  - Generate create statement at runtime ..... 43
  - Generate drop statement at runtime ..... 43
  - Generate SQL ............................ 43
  - Generate truncate statement at runtime .... 43
  - Generate UOWID ......................... 44
  - Immediate access ......................... 44
  - Insert statement ......................... 44
  - Isolation level ........................... 44
  - Key column .............................. 46
  - Null order ............................... 46
  - Limit settings ............................ 46
  - Load type ............................... 46
  - Logging ................................. 46
  - Log column values on first row error ...... 47
  - Log key values only ..................... 47
  - Log table ............................... 47
  - Logon mechanism ......................... 47
  - Macro database .......................... 47
  - Make duplicate copies .................... 47
  - Max buffer size ......................... 48
  - Max partition sessions .................... 48
  - Max sessions ............................ 48
  - Maximum bytes per character ............ 49
  - Min sessions ............................ 49
  - Missing delete rows ..................... 49
  - Missing update rows ..................... 49
  - Missing update rows ..................... 49
Chapter 7. Teradata Multiload stage 111
Introduction to Teradata MultiLoad/TPump/FastExport Stage ............ 111
FastExport Stage ........................................ 111
Input Links ............................................... 111
Output Links ............................................. 112
Installing the Teradata MultiLoad/TPump/FastExport Stage .......... 112
Configuring the Environment for Teradata Stages .......................... 112
Teradata Stages and the Parallel Canvas .................................. 112
Mapping String Data ....................................... 113
Functionality of Teradata MultiLoad/TPump/FastExport Stages ....... 113
Building a MultiLoad or TPump Script ................................ 114
Creating a InfoSphere DataStage Job to Load Data .................... 114
Defining Properties to Load Data ................................ 114
About the Stage Page .................................... 114
About The Input Page ................................... 114
Compiling and Running the Job .................................. 119
Enhancing Performance of a MultiLoad Script ........................... 120
Building a FastExport Script .................................. 120
Creating an InfoSphere DataStage Job to Export Data .................. 121
Defining Properties to Export Data ................................ 121
About the Stage Page .................................... 121
About The Output Page .................................. 121
Compiling and Running the Job .................................. 123
Writing Status Messages when Tracing Is Enabled ...................... 123
Data Type Support ......................................... 125

Chapter 8. Environment variables:
Teradata connector ........................................ 127
CC_GUARDIUM_EVENTS .................................. 127
CC_IGNORE_TIME_LENGTH_AND_SCALE ..................... 127
CC_MSG_LEVEL .......................................... 127
CC_SE_TIMESTAMP_FF .................................. 128
CC_USE_EXTERNAL_SCHEMA_ON_MISMATCH .................. 128
CC_TERA_DEBUG ........................................ 128
CC_TERA_FAIL_ON_DBS_CRASH ................................ 129
CC_TERA_MAX_DECIMAL_RETURN ................................ 129
CC_TERA_OPTIMIZE_CONNECTIONS ......................... 129
CC_TERA_RECONNECT_FLAG ................................ 129
CC_TERA_TMSM_LOG_EVENT ................................ 129
CC_TERA_USE_ISO88591_ASCII_MAP .......................... 130
CC_TRUNCATE_NSTRING_WITH_NULL .......................... 130
DS_NO_PROCESS_METADATA ................................ 130

Appendix A. Product accessibility .................................. 153
Appendix B. Reading command-line syntax ............................... 155
Appendix C. How to read syntax diagrams ............................... 157
Appendix D. Contacting IBM .................................... 159
Appendix E. Accessing and providing feedback on the product documentation ................................ 161
Notices and trademarks .................................... 163
Index ......................................................... 167
Chapter 1. Connector Migration Tool

To take advantage of the additional functionality that connectors offer, use the Connector Migration Tool to migrate jobs to use connectors instead of plug-in and operator stages.

The following table lists the stages that can be migrated to connectors and the corresponding connectors that they are migrated to:

<table>
<thead>
<tr>
<th>Stage</th>
<th>Connector</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2Z stage</td>
<td>DB2 Connector</td>
</tr>
<tr>
<td>DB2 UDB API stage</td>
<td></td>
</tr>
<tr>
<td>DB2 UDB Enterprise stage</td>
<td></td>
</tr>
<tr>
<td>DB2 UDB Load stage</td>
<td></td>
</tr>
<tr>
<td>DRS Stage</td>
<td>DRS Connector</td>
</tr>
<tr>
<td>Java Client stage</td>
<td>Java Integration Stage</td>
</tr>
<tr>
<td>Java Transformer stage</td>
<td></td>
</tr>
<tr>
<td>Netezza Enterprise stage</td>
<td>Netezza Connector</td>
</tr>
<tr>
<td>ODBC Enterprise stage</td>
<td>ODBC Connector</td>
</tr>
<tr>
<td>ODBC (Server) stage</td>
<td></td>
</tr>
<tr>
<td>SQLServer Enterprise stage</td>
<td></td>
</tr>
<tr>
<td>Oracle OCI stage</td>
<td>Oracle Connector</td>
</tr>
<tr>
<td>Oracle OCI Load stage</td>
<td></td>
</tr>
<tr>
<td>Oracle Enterprise stage</td>
<td></td>
</tr>
<tr>
<td>Teradata API stage</td>
<td>Teradata Connector</td>
</tr>
<tr>
<td>Teradata Enterprise stage</td>
<td></td>
</tr>
<tr>
<td>Teradata Load stage</td>
<td></td>
</tr>
<tr>
<td>Teradata Multiload stage</td>
<td></td>
</tr>
<tr>
<td>WebSphere® MQ stage</td>
<td>WebSphere MQ Connector</td>
</tr>
</tbody>
</table>

Migrating jobs to use connectors

To migrate jobs to use the connectors, you need to run the Connector Migration Tool.

To run the Connector Migration Tool, start it from the Microsoft Windows Programs menu or from the command line. If you start the tool from the command line, additional options that are not provided in the user interface are available.

The user interface leads you through the process of evaluating which jobs, shared containers, and stages to migrate. You select the jobs that you want to migrate, and beside each job name, the tool displays an icon that indicates whether or not the job can be fully migrated, partially migrated, or not migrated at all. To refine the list of jobs to evaluate, you can specify that only jobs that contain specific plug-in and operator stages be listed. The tool gives you a chance to make a backup of a job before you migrate it. You can make a backup copy of the job and then migrate the backup, or you can make a backup copy of the job and then migrate the original job. Either way, your original job is never lost. The job is migrated and
placed in the same folder as the original job, and the log file CCMigration.log, which records the results of the migration, is created in the current directory.

The Connector Migration Tool command line options provide the same functionality as the user interface, as well as a few additional options. Using the command line, you can perform these additional tasks:

- Specify a list of job names to be considered for migration.
- Specify a list of shared container names to be considered for migration.
- Specify a list of stage type names to limit the jobs that are considered for migration.
- Run a practice migration, where the actual migration does not take place but the possible results of the migration are placed in the log file. You can review the results and then refine the migration as necessary before you run the actual migration.
- Produce a report of jobs and their stages and stage types

Note:
- The Connector Migration Tool does not read environment variables at the operating system level. Environment variables are read only if they are defined within InfoSphere DataStage at the Project level or at the Job level. Project level environment variables are read first, then overwritten by Job environment variables. Environment variables with blank default values are ignored by the Connector Migration Tool. The default values of the environment variables are migrated, but the run-time values are not migrated.
- Throughout this documentation, the term "job" refers to parallel shared containers and server shared containers, as well as IBM® InfoSphere® DataStage® jobs.

Using the user interface to migrate jobs

Use the Connector Migration Tool to view which jobs and stages are eligible for migration and then migrate them to use connectors rather than plug-in and operator stages.

About this task

You use the same project connection details to connect to the Connector Migration Tool as you use to connect to the InfoSphere DataStage and QualityStage® Designer or InfoSphere DataStage and QualityStage Director Client. You must have sufficient user privileges to create and modify the jobs that you are migrating.

Procedure

1. Choose Start > Programs > IBM InfoSphere Information Server > Connector Migration Tool.
2. In the Log on window, complete these fields:
   a. In the Host field, enter the host name of the services tier. You can specify an optional port by separating it from the host name with a colon. The host name that you specify here is the same one that you specify when you start the Designer client, for example, mymachine:9080).
   b. In the User name field, enter your InfoSphere DataStage user name.
   c. In the Password field, enter your InfoSphere DataStage password.
   d. In the Project field, enter the name of the project. To access an InfoSphere DataStage server that is remote from the domain server, specify the project
name in full as server:[port]/project. As an alternative, you can press the button adjacent to the Project field to display a dialog box from which you can select the fully-qualified project name.

e. Click OK. An icon indicates the status of each job. A gray icon indicates that the job cannot be migrated. A gray icon with a question mark indicates that the job might be successfully migrated.

3. Display the jobs and stages to consider for migration:
   • Choose View > View all jobs to display all of the jobs in the project. This is the default view.
   • Choose View > View all migratable jobs to display all of the jobs that are in the project and that can be migrated to use connectors. Jobs that do not contain any stages that can be migrated are excluded from the job list.
   • Choose View > View jobs by stage types to open the Filter by stage type window.

4. Perform the following steps to analyze jobs:
   a. Highlight the job in the job list.
   b. Expand the job in the job list to view the stages in the job.
   c. Select one or more jobs, and click Analyze.

After analysis, the color of the job, stage, or property icon indicates whether or not it can be migrated. A green icon indicates that the job, stage, or property can be migrated. An red icon indicates that the job or stage cannot be migrated. An orange icon indicates that a job or stage can be partially migrated and that a property in a stage has no equivalent in a connector. A gray icon indicates that the job or stage is not eligible for migration.

**Note:** The Connector Migration Tool displays internal property names, rather than the names that the stages display. To view a table that contains the internal name and the corresponding display name for each property, from the IBM InfoSphere DataStage and QualityStage Designer client, open the Stage Types folder in the repository tree. Double-click the stage icon, and then click the Properties tab to view the stage properties.

5. Click Preferences and choose how to migrate the job:
   • Choose Clone and migrate cloned job to make a copy of the job and then migrate the copy. The original job remains intact.
   • Choose Back up job and migrate original job to make a copy of the job and then migrate the original job.
   • Choose Migrate original job to migrate the job without making a backup.

6. Select the jobs and stages to migrate, and then click Migrate.

The jobs and stages are migrated and are placed in the same folder as the original job. If logging is enabled, a log file that contains a report of the migration task is created. After a job is successfully migrated, a green checkmark displays beside the job name in the Jobs list to indicate that the job has been migrated.

**Using the command line to migrate jobs**

Run the Connector Migration Tool from the command line to use additional options that are not available in the user interface.
About this task

To run the Connector Migration Tool from the command line, you specify the command **CCMigration**, followed by a series of required and optional parameters. If the Connector Migration Tool is started from the command line, its user interface will be displayed if none of the options -C, -M or -B are specified. If any one of these options is specified, then the migration will proceed without any further interaction with the user. The command line options described below can therefore be used whether or not the user interface is displayed.

After a job is successfully migrated, a green checkmark displays beside the job name in the Jobs list to indicate that the job has been migrated.

Procedure

1. From the IBM InfoSphere DataStage client command line, go to the `<InformationServer>\Clients\CCMigrationTool` directory.
2. Enter the command **CCMigration**, followed by the following required parameters:
   - `-h host:port`, where *host:port* is the host name and port of the InfoSphere DataStage server. If you do not specify a port, the *port* is 9080 by default.
   - `-u user name`, where *user name* is the name of the InfoSphere DataStage user.
   - `-p password`, where *password* is the password of the InfoSphere DataStage user.
   - `-P project`, where *project* is the name of the project to connect to. To specify an InfoSphere DataStage server that is remote from the domain server, specify the fully qualified project name by using the format `server:[port]/project`.
   - One of the following:
     - `-M` If you specify this parameter, the original jobs are migrated, and backup jobs are not created.
     - `-B job name extension`, where *job name extension* is a set of alphanumeric characters and underscores. If you specify this parameter, the Connector Migration Tool creates backup jobs, names the backup jobs *source job name+job name extension*, and then migrates the original jobs. The backup jobs are saved in the same location in the repository as the source jobs.
     - `-C job name extension`, where *job name extension* is a set of alphanumeric characters and underscores. If you specify this parameter, the Connector Migration Tool clones the source jobs, names the cloned jobs *source job name+job name extension*, and then migrates the cloned jobs. The cloned jobs are saved in the same location in the repository as the source jobs.

   If you specify one of these options, the migration proceeds without requiring any additional user input. If you do not specify -M, -B, or -C, the user interface is displayed so that you can make additional choices for how to migrate the jobs.
3. Optional: Enter any of the following optional parameters:
   - `-L log file`, where *log file* is the file name and path for the log file that records the results of the migration.
   - `-S stage types`, where *stage types* is a comma-separated list of stage types. By default, the Connector Migration Tool migrates all stage types. Use this parameter to migrate only jobs that contain the specified stage types. If you specify both the `-S` and `-J` parameters, only the specified stage types within the specified jobs are migrated. If you specify the `-S` parameter and do not specify the `-C`, `-M` or `-B` parameter, only jobs that contain the specified stage
types appear in the job list that is displayed in the user interface. Limiting the jobs that are displayed can significantly reduce the startup time of the Connector Migration Tool.

- **-J job names**, where job names is a comma-separated list of jobs. By default, the Connector Migration Tool migrates all eligible jobs in the project. Use this parameter to migrate only specific jobs. If you specify the -J parameter and do not specify the -C, -M or -B parameter, only the specified jobs appear in the job list that is displayed in the user interface. Limiting the jobs that are displayed can significantly reduce the startup time of the Connector Migration Tool.

- **-c shared container names**, where shared container names is a comma-separated list of shared containers. By default, the Connector Migration Tool migrates all eligible shared containers in the project. Use this parameter to migrate only specific shared containers. If you specify the -c parameter and do not specify the -C, -M, or -B parameter, only the specified shared containers appear in the job list that displays in the user interface. Limiting the shared containers that display might significantly reduce the startup time of the Connector Migration Tool.

- **-R** If you specify this parameter, the Connector Migration Tool reports the details of the migration that would occur if the specified jobs were migrated, but does not perform an actual migration. The details are reported in the log file that is specified by using the -L parameter.

- **-a auth file**, where auth file is the file name that records the user name and password.

- **-A** If you specify this parameter, the Connector Migration Tool adds an annotation to the job design. The annotation describes the stages that were migrated, the job from which the stages were migrated, and the date of the migration.

- **-d job dump file**, where job dump file is the file name and path for a file where a list of jobs, shared containers, and stages is written. Using a job dump file is helpful when you want to determine which jobs are suitable for migration. You can use the -d parameter with the -J, -c, and -S parameters to list particular jobs, shared containers, and stage types, respectively.

- **-V** If you specify this parameter, the Connector Migration Tool specifies the target connector variant for migrated stages. The format of the list is a comma-separated list containing \{StageTypeName=Variant\}.

- **-v** If you specify this parameter with the -d command, the values of stage properties will be included in the report. If omitted, the report only contains stage names and types, but not the stage properties. This option is useful to identify jobs that have stages with certain property values. If this option is specified, then -S is ignored.

- **-T** If you specify this parameter, the Connector Migration Tool enables the variant migration mode. All connector stages found in jobs and containers whose stage type matches those listed by the -V command are modified.

- **-U** If you specify this parameter, the Connector Migration Tool enables the property upgrade migration mode. All connector stages found in jobs and containers whose properties match the conditions specified in the StageUpgrade.xml file are upgraded.
Example

The following command starts the Connector Migration Tool, connects to the project billsproject on the server dsserver as user billg, and migrates the jobs db2write and db2upsert:

CCMigration -h dsserver:9080 -u billg -p padd0ck
-J db2write,db2upsert -M

Deprecated stages

Connectors, which offer better functionality and performance, replace some stages, which have been deprecated and removed from the palette. However, you can still use the deprecated stages in jobs, and add them back to the palette.

The following stage types have been removed from palette for the parallel job canvas:

• DB2Z
• DB2® UDB API
• DB2 UDB Load
• DRS
• Dynamic RDBMS
• Java Client
• Java Transformer
• Netezza Enterprise
• ODBC Enterprise
• Oracle 7 Load
• Oracle OCI Load
• Oracle Enterprise
• Teradata API
• Teradata Enterprise
• Teradata Load
• Teradata Multiload
• WebSphere MQ

The following stage type has been removed from the palette for the server job canvas:

• Dynamic RDBMS

When you create new jobs, consider using connectors instead of the deprecated stages. The following table describes which connector to use in place of which deprecated stages:

<table>
<thead>
<tr>
<th>Deprecated stages</th>
<th>Connectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2Z</td>
<td>DB2 Connector</td>
</tr>
<tr>
<td>DB2 UDB API</td>
<td></td>
</tr>
<tr>
<td>DB2 UDB Enterprise</td>
<td></td>
</tr>
<tr>
<td>DB2 UDB Load</td>
<td></td>
</tr>
<tr>
<td>DRS</td>
<td>DRS connector</td>
</tr>
<tr>
<td>Deprecated stages</td>
<td>Connectors</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>Dynamic RDBMS</td>
<td>DB2 Connector</td>
</tr>
<tr>
<td></td>
<td>Oracle Connector</td>
</tr>
<tr>
<td></td>
<td>ODBC Connector</td>
</tr>
<tr>
<td>Java Client</td>
<td>Java Integration Stage</td>
</tr>
<tr>
<td>Java Transformer</td>
<td></td>
</tr>
<tr>
<td>Netezza Enterprise</td>
<td>Netezza connector</td>
</tr>
<tr>
<td>ODBC Enterprise</td>
<td>ODBC Connector</td>
</tr>
<tr>
<td>Oracle 7 Load</td>
<td>Oracle Connector</td>
</tr>
<tr>
<td>Oracle OCI Load</td>
<td></td>
</tr>
<tr>
<td>Oracle Enterprise</td>
<td></td>
</tr>
<tr>
<td>Teradata API</td>
<td>Teradata Connector</td>
</tr>
<tr>
<td>Teradata Enterprise</td>
<td></td>
</tr>
<tr>
<td>Teradata Load</td>
<td></td>
</tr>
<tr>
<td>Teradata Multiload</td>
<td></td>
</tr>
<tr>
<td>WebSphere MQ</td>
<td>WebSphere MQ Connector</td>
</tr>
</tbody>
</table>

To use any of the deprecated stage types in new jobs, drag the stage type from the repository tree to the canvas or to the palette. From the repository tree, expand **Stage Types.** Under **Stage Types,** expand **Parallel** or **Server** depending on the stage that you want to use. Drag the stage type to the job canvas or to the palette.
Chapter 2. Configuring access to Teradata databases

To configure access to Teradata databases, you must install database client libraries and include the path to these installed libraries in the library path environment variable. For more information about setting environment variables, see the topic about setting environment variables.

**Procedure**
1. Install database client libraries.
2. Configure access to Teradata databases.

### Configuring access to Teradata databases

To configure access to Teradata databases, you must install Teradata tools and Teradata transporters and set the environment variables.

#### Before you begin

Install database client libraries.

**Procedure**
1. Install Teradata tools and utilities on all nodes that run parallel jobs. For more information, see the installation instructions in the Teradata documentation.
2. Install the Teradata Parallel Transporter. For more information, see the Teradata product documentation.
3. Set the environment variables.

**Table 3. Required Environment variables**

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Environment variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIX</td>
<td>TWB_ROOT=/usr/tbuild/08.01.00.02</td>
</tr>
<tr>
<td></td>
<td>PATH=$TWB_ROOT/bin:$PATH</td>
</tr>
<tr>
<td></td>
<td>LIBPATH=$TWB_ROOT/lib:$LIBPATH</td>
</tr>
<tr>
<td></td>
<td>NLSPATH=$TWB_ROOT/msg/%N</td>
</tr>
<tr>
<td></td>
<td>export TWB_ROOT PATH LIBPATH NLSPATH</td>
</tr>
<tr>
<td>HP-UX</td>
<td>TWB_ROOT=/usr/tbuild/08.01.00.02</td>
</tr>
<tr>
<td></td>
<td>PATH=$TWB_ROOT/bin:$PATH</td>
</tr>
<tr>
<td></td>
<td>SHLIB_PATH=$TWB_ROOT/lib:$SHLIB_PATH</td>
</tr>
<tr>
<td></td>
<td>NLSPATH=$TWB_ROOT/msg/%N</td>
</tr>
<tr>
<td></td>
<td>export TWB_ROOT PATH SHLIB_PATH NLSPATH</td>
</tr>
<tr>
<td>Solaris</td>
<td>TWB_ROOT=/usr/tbuild/08.01.00.02</td>
</tr>
<tr>
<td></td>
<td>PATH=$TWB_ROOT/bin:$PATH</td>
</tr>
<tr>
<td></td>
<td>LD_LIBRARY_PATH=$TWB_ROOT/lib:$LD_LIBRARY_PATH</td>
</tr>
<tr>
<td></td>
<td>NLSPATH=$TWB_ROOT/msg/%N</td>
</tr>
<tr>
<td></td>
<td>export TWB_ROOT PATH LD_LIBRARY_PATH NLSPATH</td>
</tr>
</tbody>
</table>

### Testing database connections by using the ISA Lite tool

After you establish connection to the databases, test the database connection by running the IBM Support Assistant (ISA) Lite for InfoSphere Information Server tool.
For more information about the ISA Lite tool, see the topic about installation
verification and troubleshooting.

Setting the library path environment variable

To apply an environment variable to all jobs in a project, define the environment variable in the InfoSphere DataStage and QualityStage Administrator. The values that are specified for the library path and path environment variables at the project or job level are appended to the existing system values for these variables.

**About this task**

For example, suppose that directory `/opt/branded_odbc/lib` is specified as the value for the library path environment variable at the project level. Directory `/opt/IBM/InformationServer/Server/branded_odbc/lib`, which contains the same libraries but in a different location is already in the library path that is defined at the operating system level or the dsenv script. In this case, the libraries from directory `/opt/IBM/InformationServer/Server/branded_odbc/lib` are loaded when the job runs because this directory appears before directory `/opt/branded_odbc/lib` in the values that are defined for the library path environment variable.

The name of the library path environment variable depends on your operating system.

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Library path environment variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Windows</td>
<td>PATH</td>
</tr>
<tr>
<td>HP-UX</td>
<td>SHLIB_PATH</td>
</tr>
<tr>
<td>IBM AIX®</td>
<td>LIBPATH</td>
</tr>
<tr>
<td>Other supported Linux and UNIX operating systems, and HP-IA</td>
<td>LD_LIBRARY_PATH</td>
</tr>
</tbody>
</table>

On Linux or UNIX operating systems, the environment variables can be specified in the dsenv script. InfoSphere Information Server installations on Windows operating system do not include the dsenv script.

**Setting the library path environment variable in the dsenv file**

On Linux or UNIX operating systems, you can specify the library path environment variables in the dsenv script. When environment variables are specified in the dsenv script, they apply to all InfoSphere DataStage projects that run under the InfoSphere Information Server engine.

**Before you begin**

Install the client libraries.

**Procedure**

1. Log in as the root user.
2. Back up the `$ISHOME/Server/DSEngine/dsenv` script. `$ISHOME` is the InfoSphere Information Server installation directory (`/opt/IBM/InformationServer` by default).
3. Open the dsenv script.
4. Add the path to the directory that contains the client libraries to the library path environment variable.

5. To stop and start the InfoSphere Information Server engine, enter the following commands:
   ```
   bin/uv –admin –stop
   bin/uv –admin –start
   ```

6. To change directory to the ASB Agent home directory, enter the following commands:
   ```
   cd Install_directory/ASBNode/bin
   ```

7. To stop and start the ASB Agent processes, enter the following commands:
   ```
   ./NodeAgents.sh stopAgent
   ./NodeAgents.sh start
   ```

**Results**

After you restart the ASB Agent process, the InfoSphere Information Server domain services (WebSphere Application Server) take approximately a minute to register the event.

**Setting the library path environment variable in Windows**

On the Windows operating system, both the library path and `PATH` environment variables are represented by the `PATH`. For InfoSphere Information Server engine and ASB Agent processes to detect changes in the environment variables, the changes must be made at the system level and the InfoSphere Information Server engine must be restarted.

**Before you begin**

Install the client libraries.

**Procedure**

1. To edit the `PATH` system environment variable, click Environment Variable in Advance System Settings, and then select PATH.

2. Click Edit, then specify the path to the directory containing the client libraries.

3. Click OK.

4. Restart the InfoSphere Information Server engine.

5. Restart the ASB Agent processes.
Chapter 3. Teradata connector

Use the Teradata connector in parallel and server jobs to read, write, or look up data in Teradata databases.

The connector provides several capabilities that are unavailable in other Teradata stage types:
- Parallel immediate reads, writes, and lookups
- Parallel MultiLoad capability
- MultiLoad delete tasks
- Cursor lookups
- Restart capability for bulk loads
- Reject links for immediate lookups and writes
- Reject links for bulk loads
- Reject links for missing UPDATE and DELETE rows
- BLOB and CLOB data types
- Error message and row count feedback for immediate lookups and writes

If you need to invoke the external bulk utilities or read a data file that is in binary FastLoad format, use the Teradata Load stage or the Teradata MultiLoad stage. Also use the Teradata MultiLoad stage if you have custom scripts that use advanced scripting features.

Performance

The Teradata connector uses the Teradata Call-Level Interface (CLLv2) for immediate operations and the Teradata Parallel Transporter Direct API for bulk operations.

For immediate reads, writes, and lookups, the connector uses Teradata CLLv2 that provides immediate feedback on whether an operation succeeds. As a result, the connector rejects failed rows immediately rather than waiting until the end of a load. Although CLLv2 accesses data more slowly than other methods, it does not use one of the utility slots on the Teradata server.

For bulk exports and loads, the connector uses the EXPORT, LOAD, UPDATE, and STREAM operators in the Teradata Parallel Transporter Direct API. These operators export and load data faster than other methods, however, all except the STREAM operator use one of the utility slots on the Teradata server.

SQL statements

SQL statements specify how the connector selects, inserts, updates, or deletes data in Teradata tables. In a target context, SQL statements also specify how the connector creates, drops, or truncates the target table when the connector writes data.

Depending on how you configure the connector in a job, you can specify SQL statements for these situations:
Designing jobs that use the Teradata connector

When you design a parallel or a server job, you can configure the Teradata connector as a source, as a target, or a parallel job in a lookup context.

As a source, the Teradata connector has a single output link. You can use the connector to perform the following tasks:

- Immediate reads of data by using the Teradata Call-Level Interface (CLI)
- Bulk exports of data by using the EXPORT operator in the Teradata Parallel Transporter Direct API

As a target, the Teradata connector has a single input link and an optional reject link. You can use the connector to perform the following tasks:

- Immediate writes of data by using the Teradata CLI
- Bulk loads of data by using the LOAD, UPDATE, or STREAM operators in the Teradata Parallel Transporter Direct API

In a lookup context, the Teradata connector has a single reference link to a Lookup stage and an optional reject link. You can use the connector to perform the following task:

- Immediate lookups of data by using the Teradata CLI

Operators in the Teradata Parallel Transporter

When you extract or load data in bulk, the connector uses the operators in the Teradata Parallel Transporter. These operators replace the functionality in the Teradata FastExport, FastLoad, MultiLoad, and TPump utilities.

The following table compares the operators in the Teradata Parallel Transporter to the Teradata utilities that are used by the Teradata Load and Teradata MultiLoad stages. When you configure the connector as a target, consider the advantages and disadvantages of each operator before you select a load method.
<table>
<thead>
<tr>
<th>Operator</th>
<th>Equivalent Teradata utility</th>
<th>Purpose</th>
<th>Operator advantages</th>
<th>Operator disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXPORT</td>
<td>FastExport</td>
<td>Parallel bulk export of high-volume data.</td>
<td>• Fastest way to export high-volume data.</td>
<td>• Does not support single-AMP (Access Module Processor) SELECT statements. Such statements return one row, for example, a SELECT statement with an equality condition on the primary index or unique secondary index columns. • Uses one of the utility slots on the Teradata server.</td>
</tr>
<tr>
<td>LOAD</td>
<td>FastLoad</td>
<td>Parallel bulk INSERT of high-volume data.</td>
<td>• Fastest way to load high-volume data.</td>
<td>• Does not support UPDATE or DELETE. • Locks the table during the load. • Cannot load tables that have secondary indexes. • Cannot load views. • Uses one of the utility slots on the Teradata server.</td>
</tr>
<tr>
<td>UPDATE</td>
<td>MultiLoad</td>
<td>Parallel bulk INSERT, UPDATE, or DELETE of high-volume data.</td>
<td>• Loads high-volume data faster than the STREAM operator. • Supports INSERT, UPDATE, and DELETE. • Loads tables that have non-unique secondary indexes. • Loads views.</td>
<td>• Loads high-volume data slower than the LOAD operator. • Locks the table during the load. • Cannot load tables that have unique secondary indexes. • A stopped load can make it difficult to regain access to the target table. • Uses one of the utility slots on the Teradata server.</td>
</tr>
</tbody>
</table>
Table 4. Operators in the Teradata Parallel Transporter (continued)

<table>
<thead>
<tr>
<th>Operator</th>
<th>Equivalent Teradata utility</th>
<th>Purpose</th>
<th>Operator advantages</th>
<th>Operator disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>STREAM</td>
<td>T Pump</td>
<td>Parallel bulk INSERT, UPDATE, or DELETE of low-volume data.</td>
<td>• Supports INSERT, UPDATE, and DELETE.</td>
<td>• Loads high-volume data slower than the UPDATE operator.</td>
</tr>
</tbody>
</table>

Configuring the connector as a source

To configure the Teradata connector as a source, you must define the connection to a Teradata database, specify the properties for the output link, and define columns for the data that the connector will read.

About this task

To configure the connector as a source:

Procedure

1. Open the Teradata connector stage editor.
2. On the Properties tab, define the connection properties for the Teradata database.
3. Optional: On the Advanced tab, change the processing settings.
4. Specify information about the output link:
   a. On the Output tab, select the output link.
   b. On the Properties tab, define the usage properties for the link.
   c. On the Columns tab, define the column metadata for the link.
   d. Optional: On the Advanced tab, change the buffer settings for the link.
5. Click OK to save your changes and to close the Teradata connector stage editor.

Configuring the connector as a target

To configure the Teradata connector as a target, you must define the connection to a Teradata database, specify the properties for the input link, and define columns for the data that the connector will write.
About this task

To configure the connector as a target:

Procedure

1. Open the Teradata connector stage editor.
2. On the Properties tab, define the connection properties for the Teradata database.
3. Optional: On the Advanced tab, change the processing settings.
4. Specify information about the input link:
   a. On the Input tab, select the input link.
   b. On the Properties tab, define the usage properties for the link.
   c. On the Columns tab, define the column metadata for the link.
   d. Optional: On the Advanced tab, change the buffer settings for the link.
   e. Optional: On the Partitioning tab, change the partition settings for the link.
5. If the connector has a reject link, specify how to send data to this link:
   a. On the Reject tab, select the reject link, then define the reject conditions for the link.
   b. Optional: On the Advanced tab, change the buffer settings for the link.
6. Click OK to save your changes and to close the Teradata connector stage editor.

Load types for append operations

The Teradata connector provides high-speed load options that have different advantages for append operations: Load or Update.

When you use the connector to append rows to a table, you can set the Load type property to Load or Update. Both load types use a utility slot on the Teradata server, and both lock the target table. The Load option is usually faster, but the Update option supports partial append operations.

With the Load option, the connector uses the LOAD operator in the Teradata Parallel Transporter. The LOAD operator loads only empty tables. To perform an append operation, the connector must first load the data into the temporary work table. Then, the connector performs an INSERT...SELECT from the work table to the target table. INSERT...SELECT is an atomic operation, so either the whole statement succeeds or it fails. If any row in the work table cannot be inserted into the target table due to a unique constraint violation, then none of the rows are appended to the target table and all of the rows are rejected.

In situations where an append operation might fail due to a unique constraint violation, set the Load type property to Update unless you want the entire append operation to fail.

Avoiding deadlocks

To avoid deadlocks when the Teradata connector performs immediate writes in parallel mode, specify a partitioning method that ensures that the updates to a row come from the same partition.

About this task

Each partition uses a separate connection to the Teradata database. If multiple updates for a row do not come from the same connection, deadlocks can occur. For example, if you have a two-node configuration and you specify a round-robin
partitioning method, each partition might have updates to the same row. When the first partition updates a row, the connection for the partition holds a write lock on the row until the transaction is committed at the end of the job. The second partition cannot update the row until the connection for the first partition is released, which causes a deadlock.

To avoid deadlocks:

**Procedure**
1. Select the input link in the navigator.
2. Click the **Partitioning** tab.
3. In the **Partition type** field, select **Hash**.
4. In the **Available** list, select a key column according to the type of SQL statement for the link:
   - For an UPDATE or a DELETE statement, select the key in the WHERE clause of the statement.
   - For an INSERT statement, select the primary key of the table.

**Results**

The rows are hashed into partitions based on the value of the key column. The partitions ensure that the same partition contains all updates for a particular row.

**Configuring the Logon mechanism**

Teradata database can be configured to use the client system security directory for user authentication and authorization. Directory users are defined in a directory that is not part of Teradata database. The Directory users logon using the LDAP mechanism and their directory user names and passwords. To configure how users log on to the Teradata database, you must define the Logon mechanism property in the Teradata database and then also define the column for the data that the connector reads from.

**Procedure**
1. Start the IBM InfoSphere DataStage and QualityStage Designer client.
2. In the **Repository** pane, right-click the **Jobs** folder, and select **New > Parallel job**.
3. From the **Database** section of the palette, drag a Teradata connector to the canvas.
4. Perform the following steps to configure the Logon mechanism:
   a. Double-click the connector to open the properties.
   b. On the **Properties** tab, for the **Logon mechanism** property, change the value from TD2 to LDAP.
   c. Set the client version.
5. Click OK to save your changes and close the Teradata connector stage editor.

**Configuring the connector to perform lookups**

To configure the Teradata connector to perform a lookup, you must define the connection to a Teradata database, specify the properties for the reference link to the Lookup stage, and define columns for the data that the connector will request.

**About this task**

To configure the connector to perform a lookup:

**Procedure**

1. Open the Teradata connector stage editor.
2. On the **Properties** tab, define the connection properties for the Teradata database.
3. Optional: On the **Advanced** tab, change the processing settings.
4. Specify information about the reference link:
   a. On the **Output** tab, select the reference link.
   b. In the **Lookup Type** field, select the lookup type. You can use the sparse lookup method only in parallel jobs.
   c. On the **Properties** tab, define the usage properties for the link.
d. On the **Columns** tab, define the column metadata for the link.
e. Optional: On the **Advanced** tab, change the buffer settings for the link.

5. If the connector has a reject link, specify how to send data to this link:
   a. On the **Reject** tab, select the reject link, then define the reject conditions for the link.
   b. Optional: On the **Advanced** tab, change the buffer settings for the link.

6. Click **OK** to save your changes and to close the Teradata connector stage editor.

---

### Generating SQL statements in the connector at design time

You can configure the connector to generate SQL statements at design time in their statement properties.

**Before you begin**

Create a job that includes a connector as a source or target.

**About this task**

You can generate the SQL statement text only for those statement properties that have the **Generate SQL statement** option in the Build list.

**Note:** Under some circumstances, the connector requires a connection to generate SQL statements. When a user name and password are not supplied and a connection is required, a connection is made by using the user who is running the ASB Agent service.

**Procedure**

1. Double-click the connector on the job canvas to open the stage editor.
2. In the navigator, click the output or input link, depending on the type of job that you create.
3. Set **Generate SQL at runtime** to **No**.
4. In the **Table name** property, type the name of the table for the SQL statement.
5. For jobs in target context (input links), select the type of statement you want to generate in the **Write mode** property.
6. On the **Columns** page, define the columns to use in the SQL statement.
7. Click the **Properties** tab.
8. Click the **Build** button that is associated with the statement property, and select **Generate SQL statement** from the list.

   **Note:** The **Generate SQL statement** option will only be available for statements which that connector supports generating at design time. In some cases a connector may only support generating the SQL at runtime during job execution.

9. Click **OK** to save the job.

---

### Validating SQL statements in the connector at design time

After you generate or write a SQL statement, you can validate the statement during job design.
**About this task**

You can validate the SQL statement text only for those statement properties that have the **Validate SQL** option in the Build list.

**Note:** Under some circumstances, the connector requires a connection to validate SQL statements. When a user name and password are not supplied and a connection is required, a connection is made by using the user who is running the ASB Agent service.

**Procedure**

1. Save the job.
2. Click the **Build** button that is associated with the statement property, and select **Validate SQL**. The **Validate SQL** option is enabled only if the statement property contains a value and this option will only be available for statements which the target RDBMS supports validating.

**Results**

The connector validates the SQL statement by preparing the statement with the RDBMS it supports. If the SQL contains error, an error message is shown.

**Data types**

The Teradata connector supports most Teradata data types except INTERVAL, TIME WITH TIME ZONE, TIMESTAMP WITH TIME ZONE, and UDT.

If you import a Teradata table definition into the repository, then the column data types are automatically set to the appropriate IBM InfoSphere DataStage data type during the import. You can load the columns on the Columns tab as is. The following table lists the supported Teradata data types and their corresponding data types in InfoSphere DataStage:

<table>
<thead>
<tr>
<th>Teradata data type</th>
<th>InfoSphere DataStage data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOB</td>
<td>LongVarBinary</td>
</tr>
<tr>
<td>BYTE</td>
<td>Binary</td>
</tr>
<tr>
<td>BYTEINT</td>
<td>TinyInt</td>
</tr>
<tr>
<td>CHAR</td>
<td>Char</td>
</tr>
<tr>
<td>CLOB</td>
<td>LongVarChar</td>
</tr>
<tr>
<td>DATE</td>
<td>Date</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>Decimal</td>
</tr>
<tr>
<td>DOUBLE PRECISION</td>
<td>Double</td>
</tr>
<tr>
<td>FLOAT</td>
<td>Float</td>
</tr>
<tr>
<td>INTEGER</td>
<td>Integer</td>
</tr>
<tr>
<td>LONG VARCHAR</td>
<td>VarChar</td>
</tr>
<tr>
<td>NUMERIC</td>
<td>Numeric</td>
</tr>
<tr>
<td>REAL</td>
<td>Real</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>SmallInt</td>
</tr>
<tr>
<td>TIME</td>
<td>Time</td>
</tr>
<tr>
<td>_TIMESTAMP</td>
<td>Timestamp</td>
</tr>
<tr>
<td>Teradata data type</td>
<td>InfoSphere DataStage data type</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>VARBYTE</td>
<td>VarBinary</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>VarChar</td>
</tr>
</tbody>
</table>

If you type the columns manually on the Columns tab, or if the column definitions come from the output link of the previous stage in the job, they cannot contain any of the following InfoSphere DataStage data types:

- BigInt
- Bit
- Unknown

If you use an unsupported data type, the connector issues an error at run time.

**Note:** Since InfoSphere DataStage uses a 4-byte float and Teradata uses an 8-byte float, there can be a loss of precision during conversion. Use the InfoSphere DataStage Double data type in your job to correspond with the Teradata Float data type. This data type change will eliminate any precision loss.

## Parallel synchronization table

When the connector runs in parallel mode, it uses the parallel synchronization table to coordinate the player processes. When the connector runs in sequential mode, you can use the table to log execution statistics.

The schema of the parallel synchronization table is described in the following table. The unique primary index of the table consists of the SyncID, PartitionNo, and StartTime columns.

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SyncID</td>
<td>VARCHAR(255)</td>
<td>Identifier that is unique to the job instance and usage</td>
</tr>
<tr>
<td></td>
<td>CHARACTER SET UNICODE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOT NULL</td>
<td></td>
</tr>
<tr>
<td>PartitionNo</td>
<td>INTEGER NOT NULL</td>
<td>Partition number</td>
</tr>
<tr>
<td>StartTime</td>
<td>TIMESTAMP(6) NOT NULL</td>
<td>Start timestamp on the synchronization table server</td>
</tr>
<tr>
<td>EndTime</td>
<td>TIMESTAMP(6)</td>
<td>End timestamp on the synchronization table server</td>
</tr>
<tr>
<td>WaitTime</td>
<td>INTEGER NOT NULL</td>
<td>Total number of seconds that the instance waits for the other instances</td>
</tr>
<tr>
<td></td>
<td></td>
<td>during parallel synchronization</td>
</tr>
<tr>
<td>Column</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>OpType</td>
<td>CHAR</td>
<td>R    Immediate read (source or lookup)</td>
</tr>
<tr>
<td></td>
<td>CHARACTER SET LATIN</td>
<td>E    Bulk export</td>
</tr>
<tr>
<td></td>
<td>NOT NULL</td>
<td>W    Immediate write</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L    Bulk load</td>
</tr>
<tr>
<td></td>
<td></td>
<td>U    Bulk update</td>
</tr>
<tr>
<td></td>
<td></td>
<td>S    Bulk stream</td>
</tr>
<tr>
<td>ControlState</td>
<td>INTEGER NOT NULL</td>
<td>State counter internal to the connector</td>
</tr>
<tr>
<td>EventCount</td>
<td>INTEGER NOT NULL</td>
<td>Event counter internal to the connector</td>
</tr>
<tr>
<td>LogonSequenceNo</td>
<td>INTEGER NOT NULL</td>
<td>Logon sequence number for the control session</td>
</tr>
<tr>
<td>UserName</td>
<td>VARCHAR(30)</td>
<td>User name that performs the read or write</td>
</tr>
<tr>
<td></td>
<td>CHARACTER SET UNICODE</td>
<td>Database name</td>
</tr>
<tr>
<td>DatabaseName</td>
<td>VARCHAR(30)</td>
<td>Target table or source tables</td>
</tr>
<tr>
<td></td>
<td>CHARACTER SET UNICODE</td>
<td>Number of nodes</td>
</tr>
<tr>
<td>TableList</td>
<td>VARCHAR(1024)</td>
<td>Number of partitions</td>
</tr>
<tr>
<td>NodeCount</td>
<td>INTEGER NOT NULL</td>
<td>Total number of sessions that perform the read or write</td>
</tr>
<tr>
<td>PartitionCount</td>
<td>INTEGER NOT NULL</td>
<td>Maximum number of rows in each buffer</td>
</tr>
<tr>
<td>SessionCount</td>
<td>INTEGER NOT NULL</td>
<td>Buffer size that is based on the row length and the array size</td>
</tr>
<tr>
<td>ArraySize</td>
<td>INTEGER NOT NULL</td>
<td>Maximum allowed buffer size</td>
</tr>
<tr>
<td>BufferSize</td>
<td></td>
<td>Termination code</td>
</tr>
<tr>
<td>MaxBufferSize</td>
<td>INTEGER NOT NULL</td>
<td>Number of reads (source or lookup context) or writes (target context), where each read or write can have multiple rows depending on the array size</td>
</tr>
<tr>
<td>ExitCode</td>
<td>SMALLINT</td>
<td>Number of input rows</td>
</tr>
<tr>
<td>BlockCount</td>
<td>DECIMAL(18) NOT NULL</td>
<td>Number of reads (source or lookup context) or writes (target context), where each read or write can have multiple rows depending on the array size</td>
</tr>
<tr>
<td>InputRowCount</td>
<td>DECIMAL(18) NOT NULL</td>
<td>Number of input rows</td>
</tr>
<tr>
<td>Column</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ByteCount</td>
<td>FLOAT NOT NULL</td>
<td>Number of bytes that are read (source context), written (target context), or both (lookup context)</td>
</tr>
<tr>
<td>CheckpointState</td>
<td>INTEGER NOT NULL</td>
<td>Checkpoint state counter internal to the connector</td>
</tr>
<tr>
<td>CheckpointCount</td>
<td>DECIMAL(18) NOT NULL</td>
<td>Number of checkpoints that are issued by the instance</td>
</tr>
<tr>
<td>CheckpointRowCount</td>
<td>DECIMAL(18) NOT NULL</td>
<td>Number of rows that are checkpointed by the instance</td>
</tr>
<tr>
<td>CheckpointErrorCount1</td>
<td>DECIMAL(18) NOT NULL</td>
<td>Number of rows in Error table 1 at the time of the last checkpoint</td>
</tr>
<tr>
<td>CheckpointData</td>
<td>BLOB</td>
<td>Checkpoint data internal to the Teradata Parallel Transporter</td>
</tr>
<tr>
<td>InsertCount</td>
<td>DECIMAL(18) NOT NULL</td>
<td>Number of rows that are inserted</td>
</tr>
<tr>
<td>UpdateCount</td>
<td>DECIMAL(18) NOT NULL</td>
<td>Number of rows that are updated</td>
</tr>
<tr>
<td>DeleteCount</td>
<td>DECIMAL(18) NOT NULL</td>
<td>Number of rows that are deleted</td>
</tr>
<tr>
<td>SelectCount</td>
<td>DECIMAL(18) NOT NULL</td>
<td>Number of rows that are selected</td>
</tr>
<tr>
<td>RejectCount</td>
<td>DECIMAL(18) NOT NULL</td>
<td>Number of rows that are rejected</td>
</tr>
<tr>
<td>MissingRowCount</td>
<td>DECIMAL(18) NOT NULL</td>
<td>Number of rows that are not found</td>
</tr>
<tr>
<td>ErrorCount1</td>
<td>DECIMAL(18) NOT NULL</td>
<td>Number of rows that are sent to Error table 1</td>
</tr>
<tr>
<td>ErrorCount2</td>
<td>DECIMAL(18) NOT NULL</td>
<td>Number of rows that are sent to Error table 2</td>
</tr>
<tr>
<td>DuplicateKeyCount</td>
<td>DECIMAL(18) NOT NULL</td>
<td>Number of rows in the work table that fail to make it into the target table due to unique constraint violations</td>
</tr>
<tr>
<td>TELINFOArea</td>
<td>BLOB</td>
<td>Synchronization data internal to the Teradata Parallel Transporter</td>
</tr>
</tbody>
</table>
Teradata Multi-System Manager Integration and Dual Load

The Teradata Multi-System Manager (TMSM) is a product used to monitor, control, and administer multi-system Teradata environments. Applications can send events to the TMSM servers and these events can be viewed through the Teradata Viewpoint Portal.

Teradata Dual Load is a mechanism used to synchronize high volumes of data across multiple Teradata installations with a low response time. In a Dual Load, multiple Teradata systems are updated simultaneously with the same source data. Dual Load supports active-active solutions, where two Teradata systems run as a backup for each other as well as balance query load. In these configurations, both systems must be updated in parallel and kept synchronized.

IBM DataStage and the Teradata connector can be used to dual load into Teradata databases. You can design your DataStage jobs to use error handling and failure and recovery capabilities as part of the dual load process. For monitoring and reporting the dual load process, you can configure the Teradata connector to report progress and send job events and alerts to a TMSM server. You can access and view these events and alerts in the Teradata Viewpoint Portal.

Installation and configuration requirements for the TMSM integration

Before you create a job that uses the Teradata connector with the Teradata Multi-System Manager (TMSM) confirm the system requirements and set the required environment variables.

Confirm that the TMSM client has been installed on the DataStage engine tier. The TMSM client is installed as a separate package on top of the standard Teradata client tools and Utilities and Teradata Parallel Transporter.

The TMSM client requires the following environment variables to be configured during installation:

**TMSM_HOME**
Set to the Teradata TMSM client installation directory during installation. Verify that this environment variable specifies the installation directory of the TMSM package.

**Library path**
Include the **TMSM_HOME/lib** directory, which is where the TMSM client libraries are located, in the library path environment variable. The actual name of this library path environment variable is **PATH** on Microsoft Windows, **SHLIB_PATH** on HP-UX for PA-RISC, **LIBPATH** on IBM AIX, and **LD_LIBRARY_PATH** on all other supported UNIX and Linux operating systems.

**TMSM_ENABLED**
Specifies if event messaging is enabled or disabled. If the environment variable is either not set or set to an empty string, the TMSM event messaging is disabled. To enable TMSM event messages, set this environment variable to the path where the TMSM client is installed.

The **apiconfig.xml** file in the **TMSM_HOME** directory must define the correct host name and port of the TMSM event receiver. Also, this file identifies the data store directory, which must be accessible to store the events, if the event receiver is unavailable.
The `jmsTransportProvider.properties` file defines the ActiveMQ JMS properties. Verify that the `brokerURL` property specifies the correct TMSM event receiver.

**Sample job with TMSM integration in the Teradata connector**

You can design a job that uses the Teradata connector to send events to a Teradata Multi-System Manager server.

**Before you begin**

Before you start creating the sample job:
- Verify that the Teradata connector is installed and configured properly.
- Verify the Teradata client and Teradata Multi-System Manager (TMSM) client are installed and configured properly.

**About this task**

The example job uses the Teradata connector to send events to a TMSM. In this example job, the ODBC connector reads the data from a database and transfers that data to a Transformer stage, which then transforms that data and transfers it to the Teradata connector. The Teradata connector writes the data to the Teradata ecosystem and sends monitoring and logging events to a TMSM. If any errors occur during the transfer to the Teradata ecosystem, the Teradata connector logs an alert event in the TMSM. The connector allows only sending events to a TMSM when the connector is used for loading or writing data to the database.

The following figure shows the sample design which supports TMSM integration and uses the TMSM options in the Teradata connector.

![Diagram showing sample design with ODBC, Transformer, and Teradata connectors]

**Procedure**

1. Start the IBM InfoSphere DataStage and QualityStage Designer client.
2. In the **Repository** pane, right-click the **Jobs** folder, and select **New > Parallel job**.
3. Open the **Database** section of the palette, and drag an ODBC connector to the canvas.
4. Open the **Processing** section of the palette, and drag the Transformer stage to the canvas.
5. From the **Database** section of the palette, drag a Teradata connector to the canvas.
6. Define an ODBC connection and column definitions on each of the links between the connectors and the stage.
7. Perform the following steps to configure the Teradata connector for TMSM integration:
   a. Double-click the connector to open the properties.
   b. On the **Properties** tab, provide values for these fields:
8. Set the appropriate properties for the load or write process. For more information, see the topic, "Configuring the connector as a target" on page 16.

9. By default, **TMSM event options** is set to No. To enable TMSM integration, set this property to Yes. This enables the properties, **Generate UOW ID**, **UOW ID**, **UOW class**, and **UOW source**.

10. Click **OK** to save your changes and close the Teradata connector stage editor.

**Teradata connector support for TMSM**

After the Teradata connector is configured to send events and alerts to the TMSM, you can monitor and view reports of the dual load process using the Teradata Viewpoint Portal.

The Teradata connector supports the TMSM by:

- Reporting DataStage job events to the TMSM
- Reporting the content of DataStage errors and alert events
- Logging TMSM events in the Director

To report progress to the TMSM event listener, the connector sends the events during job execution. The following are the DataStage job events that are reported by Teradata connector to the TMSM:

- Logon
- BeforeSQL_Executed
- BeforeNodeSQL_Executed
- AfterNodeSQL_Executed
- AfterSQL_Executed
- Commit_Executed
- Rollback_Executed
- Insert_Completed
- Update_Completed
- Delete_Completed
- InsertThenUpdate_Completed
- DeleteThenInsert_Completed
- UserDefinedSQL_Completed
- Process_Report
- Checkpoint_Executed
- EndAquisition_Executed
- ApplyRows_Executed
- Logoff

### Table 5. Configuring the properties

<table>
<thead>
<tr>
<th>Field name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teradata client version</td>
<td>The <strong>TMSM event options</strong> field is only available when using version 12.</td>
</tr>
<tr>
<td>Server</td>
<td>Enter the server name. For example, type TD12.</td>
</tr>
<tr>
<td>User</td>
<td>Enter the user name.</td>
</tr>
<tr>
<td>Password</td>
<td>Enter the password.</td>
</tr>
</tbody>
</table>
When the connector encounters an unrecoverable error, it sends an alert event to the TMSM. For RDBMS errors detected in the connector, the alert code sent to the TMSM is 800000 plus the RDBMS code returned by Teradata, along with a brief description of the error. For errors specific to DataStage and job configuration, the alert code 50001 is used. In some cases, you may have to open the DataStage Director to the dual load job to view the full error description. For all alert events the connector includes the job step at which the error occurred to verify progress and allow for comparing dual load process synchronization.

By default, the connector does not log the events being sent to the TMSM in the DataStage Director log. When a connector job is configured to send events to the TMSM, it is done transparently. To see the events in the job execution logged in the DataStage Director, the environment variable `CC_TERA_TMSM_LOG_EVENT` can be used. The value of the environment variable can be set to 0 or 1. If the value is set to 1, the connector logs each job step sent to the TMSM in the DataStage Director log. If the value is set to 0, empty string, or unset, the connector holds default behavior.

### Options for client character set in server stages

The options for **Client character set** are:
- Default
- Auto
- ASCII (Unix and Windows)
- EBCDIC (IBM)
- EBCDIC037_0E (IBM U.S. and Canada)
- EBCDIC273_0E (IBM Austria and Germany)
- EBCDIC277_0E (IBM Denmark and Norway)
- HANGULEBCDIC933_1II (Korean IBM)
- HANGULKSC5601_2R4 (Korean Unix and Windows)
- KANJIEBCDIC5026_0I (Japanese IBM)
- KANJIEUC_0U (Japanese Unix)
- KANJISJIS_0S (Japanese Windows)
- LATIN1_0A (ISO 8859-1 Latin 1 Unix)
- LATIN9_0A (ISO 8859-15 Latin 9 Unix)
- LATIN1252_0A (Latin Windows)
- SCHGB2312_1T0 (Simplified Chinese Unix and Windows)
- CHBIG5_1R0 (Traditional Chinese Unix and Windows)
- TCHEBCDIC937_3IB (Traditional Chinese IBM)
- UTF8 (Universal Transformation Format)

If **Database vendor** is a job parameter, the Teradata-specific client character sets do not appear in the list of available characters sets. Only **Default** and **Auto** appear in the list. However, it is still possible to specify a Teradata-specific client character set or a job parameter in the field. This field has no effect for other vendors.

**Default** specifies that the installed Teradata client's default character set should be used to connect to the Teradata server.

**Auto** specifies that the Teradata client character set should be set automatically based on the stage map on the **NLS** tab as shown in the following table.
Table 6. Settings of client character set based on settings in stage maps

<table>
<thead>
<tr>
<th>Server Job NLS Stage Map</th>
<th>Parallel Job NLS Stage Map</th>
<th>Teradata Client Character Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASCII</td>
<td>ASCL_ASCII</td>
<td>ASCII</td>
</tr>
<tr>
<td>BIG5</td>
<td>ASCL_BIG5</td>
<td>TCHBIG5_1R0</td>
</tr>
<tr>
<td>EBCDIC</td>
<td>ASCL_EBCDIC</td>
<td>EBCDIC</td>
</tr>
<tr>
<td>EBCDIC-037</td>
<td>ASCL_EBCDIC-037</td>
<td>EBCDIC037_0E</td>
</tr>
<tr>
<td>EBCDIC-1026</td>
<td>ASCL_EBCDIC-1026</td>
<td>KANJIEBCDIC1026_0I</td>
</tr>
<tr>
<td>IBM273</td>
<td>EBCDIC273_0E</td>
<td></td>
</tr>
<tr>
<td>IBM277</td>
<td>EBCDIC277_0E</td>
<td></td>
</tr>
<tr>
<td>EBCDIC-IBM933</td>
<td>ASCL_EBCDIC-IBM933</td>
<td>HANGULEBCDIC933_1II</td>
</tr>
<tr>
<td>EBCDIC-IBM937</td>
<td>ASCL_EBCDIC-IBM937</td>
<td>TCHEBCDIC937_3IB</td>
</tr>
<tr>
<td>GB2312</td>
<td>ASCL_GB2312</td>
<td>SCHGB2312_1T0</td>
</tr>
<tr>
<td>ISO8859-1</td>
<td>ASCL_ISO8859-1</td>
<td>LATIN1_0A</td>
</tr>
<tr>
<td>ISO8859-15</td>
<td>ASCL_ISO8859-15</td>
<td>LATIN9_0A</td>
</tr>
<tr>
<td>JPN-EUC</td>
<td>ASCL_JPN-EUC</td>
<td>KANJIEUC_0U</td>
</tr>
<tr>
<td>KSC5601</td>
<td>ASCL_KSC5601</td>
<td>HANGULKSC5601_2R4</td>
</tr>
<tr>
<td>MS1252</td>
<td>ASCL_MS1252</td>
<td>LATIN1252_0A</td>
</tr>
<tr>
<td>SHIFT-JIS</td>
<td>ASCL_SHIFT-JIS</td>
<td>KANJISHIFT-JIS_0S</td>
</tr>
<tr>
<td>UTF8</td>
<td>UTF-8</td>
<td>UTF8</td>
</tr>
</tbody>
</table>

ASCII, EBCDIC, and UTF8 are permanently installed Teradata client character sets and are always available. The other client character sets must be installed on the Teradata server to be usable.

Teradata connector properties

Properties specify how the Teradata connector operates in a job. The properties are different for source, target, and lookup contexts.

Information about connector properties is presented in alphabetical order for the connection and usage properties on the Properties tab.

Access method

Use this property to specify the method for table access.

The following methods are available:

**Immediate**

Specifies that the connector uses Teradata Call-Level Interface (CLIv2) for immediate table access. This method is the default. The connector receives immediate feedback on whether an operation succeeds, so failed rows are rejected immediately instead of at the end of an operation. This method is slower than bulk access, but it does not use one of the utility slots on the Teradata server. In a source context, this method causes the connector to run in sequential mode.

**Bulk**

Specifies that the connector uses Teradata Parallel Transporter Direct API for bulk table access. Bulk access is not supported if the **Write mode**
property is set to Insert then update. This method is faster than immediate access, but it uses one of the utility slots on the Teradata server.

Account
Use this property to specify the name of the user account ID for resource accounting.

If the Server, User name, and Password properties are set, click the Account button to select an account that is available to the current user.

If you leave this property blank, Teradata assigns a default value.

After SQL
Use this property to specify SQL statements to run once after all data is processed by the connector.

This property is available if the Before/After SQL property is set to Yes.

Use semicolons to separate multiple SQL statements.

For server jobs, if you enable the Fail on error property, the job stops if an error occurs. For parallel jobs, the job does not stop if an error occurs.

If the Transaction mode property is set to ANSI, the connector automatically commits the transaction after the connector runs all of the statements in the After SQL property. DDL statements must be run in their own transactions, so they must be separated by COMMIT statements. An example of such a statement is as follows:

```
COLLECT STATISTICS ON TableName INDEX IndexName1;
COMMIT;
COLLECT STATISTICS ON TableName INDEX IndexName2;
COMMIT;
```

If the Transaction mode property is set to Teradata, the Teradata server automatically begins and ends an implicit transaction for each statement. To run the statements within a single transaction, you must precede the statements with an explicit BEGIN TRANSACTION statement and follow the statements with an explicit END TRANSACTION statement. An example of such a statement is as follows:

```
BEGIN TRANSACTION;
INSERT INTO TableName2 SELECT * FROM TableName1;
INSERT INTO TableName3 SELECT * FROM TableName1;
END TRANSACTION;
```

After SQL file
Use this property to specify the name of a text file that contains SQL statements to run once after all data is processed by the connector. Include the path if the file is not in the project directory.

This property is available if the Before/After SQL property is set to Yes.
The file must be on the conductor node and must be in the client character set that you specified for the **Client character set** property. Any line terminators must be appropriate for the conductor platform. The file must use semicolons to separate multiple SQL statements.

The statements are run after any statements in the **After SQL** property.

You can use this property to run dynamic SQL. For example, in one job the connector can query the system views to determine what secondary indexes are present on a table. The job can use the information to generate COLLECT STATISTICS statements for the index columns and write them to a sequential file on the conductor node. Another job can use the **After SQL file** property to run the statements after the connector performs a bulk load.

If the **Transaction mode** property is set to ANSI, the connector automatically commits the transaction after the connector runs all of the statements in the **After SQL file** property. DDL statements must be run in their own transactions, so they must be separated by COMMIT statements. An example of such a statement is as follows:

```sql
COLLECT STATISTICS ON TableName INDEX IndexName1;
COMMIT;
COLLECT STATISTICS ON TableName INDEX IndexName2;
COMMIT;
```

If the **Transaction mode** property is set to Teradata, the Teradata server automatically begins and ends an implicit transaction for each statement. To run the statements within a single transaction, you must precede the statements with an explicit BEGIN TRANSACTION statement and follow the statements with an explicit END TRANSACTION statement. An example of such a statement is as follows:

```sql
BEGIN TRANSACTION;
INSERT INTO TableName2 SELECT * FROM TableName1;
INSERT INTO TableName3 SELECT * FROM TableName1;
END TRANSACTION;
```

### After SQL (node)

Use this property to specify SQL statements to run once on each node after all data is processed by the connector.

This property is available if the **Before/After SQL** property is set to Yes.

Use semicolons to separate multiple SQL statements.

If the **Transaction mode** property is set to ANSI, the connector automatically commits the transaction after the connector runs all of the statements in the **After SQL (node)** property. DDL statements must be run in their own transactions, so they must be separated by COMMIT statements. An example of such a statement is as follows:

```sql
COLLECT STATISTICS ON TableName INDEX IndexName1;
COMMIT;
COLLECT STATISTICS ON TableName INDEX IndexName2;
COMMIT;
```
If the Transaction mode property is set to Teradata, the Teradata server automatically begins and ends an implicit transaction for each statement. To run the statements within a single transaction, you must precede the statements with an explicit BEGIN TRANSACTION statement and follow the statements with an explicit END TRANSACTION statement. An example of such a statement is as follows:

```
BEGIN TRANSACTION;
INSERT INTO TableName2 SELECT * FROM TableName1;
INSERT INTO TableName3 SELECT * FROM TableName1;
END TRANSACTION;
```

**Allow duplicate rows**

Use this property to specify whether to allow duplicate rows in the table that is created.

The following values are available:

- **Default**: Allows duplicate rows if the Transaction mode property is set to ANSI. Prevents duplicate rows if the Transaction mode property is set to Teradata. The connector generates the CREATE TABLE statement without a SET or MULTISET qualifier.

- **Yes**: Allows duplicate rows in compliance with the ANSI standard. The connector generates the CREATE TABLE statement with a MULTISET qualifier. Duplicate rows are allowed regardless of the setting in the Transaction mode property.

- **No**: Prevents duplicate rows. The connector generates the CREATE TABLE statement with a SET qualifier. Duplicate rows are prevented regardless of the setting in the Transaction mode property.

**Array size**

Use this property to specify the maximum number of rows to transfer in each read or write operation.

If you increase the array size, you can improve the performance of write operations. However, if any row in the array fails to write due to an error, the write for the entire array fails and all rows in the array are rejected with the same error.

If the Teradata server does not support array operations, the Teradata connector buffers input rows up to the size that you specify for this property, but the connector sends the rows to the Teradata server one at a time.

The Record count property must be a multiple of the Array size property.

**Automap character encoding**

Use this property to address certain specific cases, where different maps need to be used.

The default value of this property is false. This can be set to true in which case, it skips the automatic mapping done within Teradata Common Connector (TDCC). You then specify different map and the number of bytes per character as suitable to the new map.
Before/After SQL

Use this property to specify whether to run SQL statements before or after data is processed by the connector.

For example, you might want to run SQL statements to drop or recreate secondary indexes, or to issue COLLECT STATISTICS statements to update the table statistics after a load operation.

If you set Before/After SQL to Yes, the sub-properties are available for you to use. If you set Before/After SQL to No, the sub-properties are not available to you. The default value is No.

Before SQL

Use this property to specify SQL statements to run once before any data is processed by the connector.

This property is available if the Before/After SQL property is set to Yes.

Use semicolons to separate multiple SQL statements.

If the Transaction mode property is set to ANSI, the connector automatically commits the transaction after the connector runs all of the statements in the Before SQL property. DDL statements must be run in their own transactions, so they must be separated by COMMIT statements. An example of such a statement is as follows:

```
COLLECT STATISTICS ON TableName INDEX IndexName1;
COMMIT;
COLLECT STATISTICS ON TableName INDEX IndexName2;
COMMIT;
```

If the Transaction mode property is set to Teradata, the Teradata server automatically begins and ends an implicit transaction for each statement. To run the statements within a single transaction, you must precede the statements with an explicit BEGIN TRANSACTION statement and follow the statements with an explicit END TRANSACTION statement. An example of such a statement is as follows:

```
BEGIN TRANSACTION;
INSERT INTO TableName2 SELECT * FROM TableName1;
INSERT INTO TableName3 SELECT * FROM TableName1;
END TRANSACTION;
```

Before SQL file

Use this property to specify the name of a text file that contains SQL statements to run once before any data is processed by the connector. Include the path if the file is not in the project directory.

This property is available if the Before/After SQL property is set to Yes.

The file must be on the conductor node and must be in the client character set that you specified for the Client character set property. Any line terminators must be appropriate for the conductor platform. The file must use semicolons to separate multiple SQL statements.
The statements are run after any statements in the **Before SQL** property.

You can use this property to run dynamic SQL. For example, in one job the connector can query the system views to determine what secondary indexes are present on a table. The job can use the information to generate DROP INDEX statements and write them to a sequential file on the conductor node. Another job can use the **Before SQL file** property to run the statements before the connector performs a bulk load.

If the **Transaction mode** property is set to ANSI, the connector automatically commits the transaction after the connector runs all of the statements in the **Before SQL file** property. DDL statements must be run in their own transactions, so they must be separated by COMMIT statements. An example of such a statement is as follows:

```sql
COLLECT STATISTICS ON TableName INDEX IndexName1;
COMMIT;
COLLECT STATISTICS ON TableName INDEX IndexName2;
COMMIT;
```

If the **Transaction mode** property is set to Teradata, the Teradata server automatically begins and ends an implicit transaction for each statement. To run the statements within a single transaction, you must precede the statements with an explicit BEGIN TRANSACTION statement and follow the statements with an explicit END TRANSACTION statement. An example of such a statement is as follows:

```sql
BEGIN TRANSACTION;
INSERT INTO TableName2 SELECT * FROM TableName1;
INSERT INTO TableName3 SELECT * FROM TableName1;
END TRANSACTION;
```

**Before SQL (node)**

Use this property to specify SQL statements to run once on each node before any data is processed by the connector.

This property is available if the **Before/After SQL** property is set to **Yes**.

Use semicolons to separate multiple SQL statements.

If the **Transaction mode** property is set to ANSI, the connector automatically commits the transaction after the connector runs all of the statements in the **Before SQL (node) properties. DDL statements must be run in their own transactions, so they must be separated by COMMIT statements. An example of such a statement is as follows:

```sql
COLLECT STATISTICS ON TableName INDEX IndexName1;
COMMIT;
COLLECT STATISTICS ON TableName INDEX IndexName2;
COMMIT;
```

If the **Transaction mode** property is set to Teradata, the Teradata server automatically begins and ends an implicit transaction for each statement. To run the statements within a single transaction, you must precede the statements with
an explicit BEGIN TRANSACTION statement and follow the statements with an explicit END TRANSACTION statement. An example of such a statement is as follows:

BEGIN TRANSACTION;
INSERT INTO TableName2 SELECT * FROM TableName1;
INSERT INTO TableName3 SELECT * FROM TableName1;
END TRANSACTION;

**Buffer usage**

Use this property to specify whether requests share the same buffer or use separate buffers.

This property is available if the **Write mode** property is set to Insert then update, Update then insert, Delete then insert, or User-defined SQL.

The following values are available:

- **Share**  Specifies that requests share the same buffer. This value is the default. Select this value when the SQL statements reference all or most of the same input fields. Otherwise, some unnecessary data is sent to the Teradata server for each statement.

- **Separate**  Specifies that requests use separate buffers. Select this value when the SQL statements reference none or very few of the same input fields. More time and memory are required to build separate buffers, but no unnecessary data is sent to the Teradata server for each statement.

**Bulk access**

Use the properties in this group to specify information about bulk access of the Teradata database.

This property group is available if the **Access method** property is set to **Bulk**.

**Case sensitive**

Use this property to specify whether or not text comparisons are case-sensitive.

If the connector uses multiple input links and you choose **Ordered** in the **Record ordering** field, use the **Case sensitive** field to specify whether or not text comparisons in a sort operation are case-sensitive.

**Character set**

Use this property to specify the character set representation of an SQL text file.

The name of the character set must comply with the Internet Assigned Numbers Authority (IANA) standards, such as UTF-8 or ASCII.

If you leave this property blank, the connector expects the character set representation to match the character set in the **Client character set** property.

**Checkpoint timeout**

Use this property to specify the maximum number of seconds that each instance waits for the other instances to reach the checkpoint.
This property is available if the **Parallel synchronization** property is set to **Yes** and the **Access method** property is set to **Bulk**.

If the timeout expires, the connector terminates the load with an unrecoverable timeout error. The default value of 0 specifies that the connector uses its default wait time.

**Cleanup mode**

Use this property to specify whether to drop the error tables and the work table if a load ends with an error that cannot be restarted.

The following modes are available:

- **Drop** Specifies that the connector drops the error tables and the work table if a load ends with an error that cannot be restarted. The error tables and the work table are kept if the load can be restarted. This option is the default.
- **Keep** Specifies that the connector does not drop the error tables and the work table if a load ends with an error that cannot be restarted. Normally it is not necessary to keep these tables, because rejected rows can be sent down a reject link.

**Client character set**

Use this property to specify the Teradata client character set for connections to the Teradata server.

ASCII, EBCDIC, UTF-8, and UTF-16 (V2R5.1 or later) are permanently installed and are always available. You must install the other character sets on the Teradata server before you can use them. If the **Server**, **User name**, and **Password** properties are set, click the **Client character set** button to select a client character set.

The client character set affects the size of the data buffer. For example, a UTF-8 representation can contain characters that are up to 3 bytes long. Thus the connector allocates a buffer that is larger than required if the data only contains Latin characters. You can reduce the buffer size by using one of the single-byte character sets, such as **LATIN1_0A**, if you install it on the Teradata server.

The following character sets are available:

- **UTF8** Unicode Transformation Format in multi-byte representation.
- **UTF16** Unicode Transformation Format in double-byte representation.
- **ASCII** UNIX and Windows.
- **EBCDIC** IBM. Only for channel-attached systems that run UNIX System Services.
- **EBCDIC037_0E** IBM U.S. and Canada. Only for channel-attached systems that run UNIX System Services.
- **EBCDIC273_0E** IBM Austria and Germany. Only for channel-attached systems that run UNIX System Services.
- **EBCDIC277_0E** IBM Denmark and Norway. Only for channel-attached systems that run UNIX System Services.
Columns

Use this property to select the key column or key columns that contain large objects (LOBs) that are passed by using locator or reference information.

If the schema contains LOB columns that you want to pass inline as opposed to by reference, do not specify those columns in this property. If you are passing LOBs by reference, you must also set the **Array size** property to 1.

Click **Available columns** to select from the list of available LOB columns. You can select more than one column.

Column delimiter

Use this property to specify the delimiter that you want to use between column values in the log.

This property is available when you specify **Yes** for the **Log column values on first row error** property. The **Column delimiter** property is not available when **Write mode** property is set to **Bulk load**.

The supported values for the **Column delimiter** property are Space, Newline, Tab, and Comma.
Create statement
Use this property to define a CREATE TABLE statement to create the target table.

This property is required when the Generate create statement at runtime property is set to No.

To define a CREATE TABLE statement:
• Type a statement in the cell next to the Create statement property.
• Click the browse button to open the Create Statement window, which contains a large text box where you can type a statement.
• Click the Job Parameter button to use a job parameter.
• Click the Generate button to generate the statement automatically if you specified a value for the Table name property and defined column metadata on the Columns tab. You can modify the statement if necessary.

Create table options
Use the properties in this group to specify information about a new table that you create.

This property is available if the Generate create statement at runtime property is set to Yes.

Data block size
Use this property to specify the data block size of the table in bytes.

The default value of 0 specifies that the connector generates the CREATE TABLE statement without a DATABLOCKSIZE clause.

Database
Use this property to specify the name of the default database for the Teradata connection.

If the Server, User name, and Password properties are set, click the Database button to select a database on the server.

If you leave this property blank, the connector uses the default database for the user in the User name property.

Delete multiple rows
Use this property to specify whether a delete task deletes multiple rows from a table.

This property is available if the Write mode property is set to Delete.

This property deletes multiple rows faster than a DELETE statement, which matches only one row at a time. You cannot use this property to delete from a view. If a delete task does not reference any input columns, it has no rows as input. If a delete task references input columns, it has one row as input.

The following values are available:
Yes Specifies that a delete task is used. This value causes the connector to run in sequential mode. The Delete statement property can specify a ranged
delete. If the statement references input columns, then the input data set
must have one row. An example of such a statement is as follows:

```
DELETE FROM TableName WHERE ColumnName BETWEEN
ORCHESTRATE.InputValue1 AND ORCHESTRATE.InputValue2
```

If the statement does not reference any input columns, then the input data
set must have zero rows. The statement can use job parameters, but cannot
reference any input fields. An example of such a statement is as follows:

```
DELETE FROM TableName WHERE ColumnName BETWEEN #InputValue1# AND
#InputValue2#
```

No Specifies that each input row deletes one row at a time. The `Delete
statement` property must specify a WHERE clause that contains an equality
predicate on the primary key. An example of such a statement is as
follows:

```
DELETE FROM TableName WHERE ColumnName = ORCHESTRATE.InputValue
```

**Delete statement**

Use this property to define a DELETE statement to remove rows from the
database.

This property is required when the `Generate SQL` property is set to No and when
the `Write mode` property is set to Delete or Delete then insert.

To define a DELETE statement:

- Type a statement in the cell next to the `Delete statement` property.
- Click the browse button to open the Delete Statement window, which contains a
  large text box where you can type a statement.
- Click the `Job Parameter` button to use a job parameter.
- Click the `Build` button to open the SQL Builder window, a graphical interface
  that helps you build the statement.

**Drop statement**

Use this property to define a DROP TABLE statement to drop the target table.

This property is required when the `Generate drop statement at runtime` property
is set to No.

To define a DROP TABLE statement:

- Type a statement in the cell next to the `Drop statement` property.
- Click the browse button to open the Drop Statement window, which contains a
  large text box where you can type a statement.
- Click the `Job Parameter` button to use a job parameter.
- Click the `Generate` button to generate the statement automatically if you
  specified a value for the `Table name` property and defined column metadata on
  the Columns tab. You can modify the statement if necessary.

**Duplicate insert rows**

Use this property to specify whether to reject or ignore duplicate rows if an
INSERT statement fails due to a unique constraint violation.
This property is available if the **Write mode** property is set to Insert, Update then insert, Delete then insert, or User-defined SQL.

The following values are available:

**Default**
- Specifies that the Teradata Parallel Transporter Direct API decides the default action. In most cases, the action is to reject the row.
- **Reject** Rejects duplicate rows to one of the error tables.
- **Ignore** Ignores duplicate rows. No entry to an error table is created.

**Duplicate update rows**
Use this property to specify whether to reject or ignore duplicate rows if an UPDATE statement fails due to a unique constraint violation.

This property is available if the **Write mode** property is set to Update, Update then insert, or User-defined SQL.

The following values are available:

**Default**
- Specifies that the Teradata Parallel Transporter Direct API decides the default action. In most cases, the action is to reject the row.
- **Reject** Rejects duplicate rows to one of the error tables.
- **Ignore** Ignores duplicate rows. No entry to an error table is created.

**Enable LOB references**
Use the properties in this group to specify whether large object (LOB) columns in the **Columns** property are passed through the connector by using locator information. If the schema contains LOB columns and you set **Enable LOB references** to No, all LOB columns are passed inline. If you set **Enable LOB references** to Yes, only the specified columns are passed by reference. The remaining LOB columns are passed inline.

If you set **Enable LOB references** to Yes, the value that is passed as a locator is read from the database when the target stage runs. But, the rest of the data is read when the source stage runs. You also must set **Array size** to 1 if **Enable LOB references** is Yes.

The following IBM InfoSphere DataStage data types are treated as LOBs:
- LongVarBinary for database binary large object (BLOB) columns
- LongVarChar and LongNVarChar for database character large object (CLOB) columns

The default value is No.

**Enable quoted identifiers**
Use this property to specify whether to enclose database object names in double quotation marks in the generated SQL.

Double quotation marks are required if the table or column names contain special characters. This property only applies to the **Table name** property and the columns in the generated SQL.
End of data
Use this property to specify whether to insert end-of-wave markers for the last set of records when there are fewer records than the number in the **Record count** property.

This property is available if the **End of wave** property is set to **Before** or **After**.

End of wave
Use this property to specify settings for end-of-wave handling.

Use end-of-wave markers to divide a large job into smaller transactions or "units of work." If you configure your large jobs to use end-of-wave markers, the connector commits each transaction after the transaction is successfully processed.

The following values are available:
- **None**  
  Specifies that end-of-wave markers are never inserted into the record data set. This value is the default.
- **Before**  
  Specifies that end-of-wave markers are inserted before the transaction is committed.
- **After**  
  Specifies that end-of-wave markers are inserted after the transaction is committed.

End row
Use this property to specify the row number at which to stop the load operation.

The default value of 0 specifies that processing ends with the last row. Keep the default value if the connector runs in parallel mode. If you change the default value, the connector runs in sequential mode. You must specify a value that is greater than or equal to the value of the **Start row** property.

End timeout
Use this property to specify the maximum number of seconds that each instance waits for the other instances to finish processing.

This property is available if the **Parallel synchronization** property is set to **Yes**.

The default value of 0 specifies that the connector uses its default wait time.

Error control
Use the properties in this group to specify error control information.

This property group is available if the **Load type** property is set to **Update** or **Stream**.

Error limit
Use this property to specify the maximum number of rows that can be rejected to the first error table.

The default value of 0 does not limit the number of rejected rows. If you expect no errors in the input data, specify a value of 1 to have the job terminate when any row causes an error.
Error table 1

Use this property to specify the name of the first error table for the bulk load.

Specify a value that is unique to the job instance and usage. If you leave this property blank, the default name is `table_name_ET`. Double quotation marks are required if the table name contains special characters, as defined by Teradata. To put the error table in a different database than the source or target table, qualify the table name with a database name.

This error table typically contains rows that are rejected because of an error other than a unique constraint violation.

Error table 2

Use this property to specify the name of the second error table for the bulk load.

This property is available if the `Load type` property is set to `Load` or `Update`.

Specify a value that is unique to the job instance and usage. If you leave this property blank, the default name is `table_name_UV`. Double quotation marks are required if the table name contains special characters, as defined by Teradata. To put the error table in a different database than the source or target table, qualify the table name with a database name.

This error table typically contains rows that have a unique constraint violation.

Fail on error

Use this property to specify whether to terminate the job or continue processing if the SQL statement in the parent property fails.

The following values are available:

- **Yes**: Specifies that the connector raises an unrecoverable exception and terminates the job if a statement fails. This value is the default.
- **No**: Specifies that the connector issues a warning and continues processing if a statement fails.

Fail on size mismatch

Use this property to specify whether a job fails if the sizes of numeric and string fields do not match. The size comparison occurs when the connector compares the design schema with the schema that is returned by the SQL statement.

The following values are available:

- **Yes**: Specifies that the job fails if the sizes of numeric and string fields do not match. An error message indicates which field has incompatible sizes in the design schema and the external schema.

- **No**: Specifies that the job does not fail if the sizes of numeric and string fields do not match. A warning message is written to the log to indicate which field has incompatible sizes in the design schema and the external schema. Data truncation is possible.
File

Use this property to specify the name of a text file that contains the SQL statements to run for each input row. Include the path if the file is not in the project directory.

This property is required when the **User-defined SQL** property is set to **File**.

The file must be on the conductor node and must be in the character set in the **Character set** property or the **Client character set** property if the **Character set** property is empty. Any line terminators must be appropriate for the conductor platform.

You can use this property to run dynamic SQL for each input row.

If you want to send row counts to the reject link, you can use this property to run dynamic SQL instead of using the **Before SQL file** or **After SQL file** properties.

Free space percent

Use this property to specify the percentage of free space that remains on a cylinder during load or update operations.

The **Table free space** property must be set to **Yes**.

To set this property, type a number from 0 to 75.

Generate create statement at runtime

Use this property to specify whether the connector generates the CREATE TABLE statement at run time.

This property is available if the **Table action** property is set to **Create** or **Replace**.

The default value is **Yes**. If you set this property to **No**, then you must supply the CREATE TABLE statement by using the **Create statement** property.

Generate drop statement at runtime

Use this property to specify whether the connector generates the DROP TABLE statement at run time.

This property is available if the **Table action** property is set to **Replace**.

The default value is **Yes**. If you set this property to **No**, then you must supply the DROP TABLE statement by using the **Drop statement** property.

Generate SQL

Use this property to specify whether the connector generates the SELECT, INSERT, UPDATE, or DELETE statement at run time.

The default value is **No**. If you set this property to **Yes**, then you must supply the table name by using the **Table name** property.

Generate truncate statement at runtime

Use this property to specify whether the connector generates the DELETE statement at run time.
This property is available if the **Table action** property is set to **Truncate**.

The default value is **Yes**. If you set this property to **No**, then you must supply the DELETE statement by using the **Truncate statement** property.

**Generate UOWID**

Use this property to specify whether the connector generates the Unit of Work ID (UOWID) at run time.

Select **Yes** to have the connector automatically generate the UOW ID for the Teradata Multi-System Manager (TMSM) at run time. Select **No** to have a user-defined UOW ID.

This property is available when the **TMSM event options** property is set to **Yes**. The default value is **No**.

If **Generate UOWID** property is set to **Yes**, the connector generates a globally unique identifier (GUID) with format `<DataStage project>-<DataStage job>-<Current timestamp>` for the UOWID. If the generated UOWID is greater than 50 characters, which is the maximum length allowed by Teradata Multi-System Manager (TMSM), then the connector specifies the format, `<DataStage job>-<Current timestamp>`, or `<DataStage project>-<Current timestamp>`, or `“DataStage”-<Current timestamp>`.

If you set this property to **No**, the **UOW ID** property will be enabled to allow the user to define the TMSM_UOWID.

**Immediate access**

Use the properties in this group to specify information for the immediate access of the Teradata database.

This property group is available if the **Access method** property is set to **Immediate**.

**Insert statement**

Use this property to define an INSERT statement to insert rows into the database.

This property is required when the **Generate SQL** property is set to **No** and when the **Write mode** property is set to **Insert**, **Insert then update**, **Update then insert**, or **Delete then insert**.

To define an INSERT statement:

- Type a statement in the cell next to the **Insert statement** property.
- Click the browse button to open the Insert Statement window, which contains a large text box where you can type a statement.
- Click the **Job Parameter** button to use a job parameter.
- Click the **Build** button to open the SQL Builder window, a graphical interface that helps you build the statement.

**Isolation level**

Use this property to specify the degree of isolation that an application process has from the concurrent application processes. By specifying whether to lock rows, you can control how transactions are committed.
When the connector performs a lookup, this property is valid within the scope of the current transaction. If an end-of-wave event occurs in the middle of a job, the commit releases the read lock on any rows that were read, and those reads are not repeatable.

If an error occurs on a lookup when the Transaction mode property is set to Teradata, Teradata automatically rolls back the current transaction. A rollback releases the read lock on any rows that were read, and those reads are not repeatable. If an error occurs on a lookup when the Transaction mode property is set to Teradata and the row is rejected, the connector issues a warning and begins a new transaction. If the Transaction mode property is set to ANSI, an error will not affect the current transaction.

The following levels are available:

**Default**
Specifies that the connector prefixes the SELECT statement without a locking modifier. The query uses the default isolation level for the connection. This value is the default.

**Read uncommitted**
Specifies that the connector prefixes the SELECT statement with a LOCK ROW FOR ACCESS modifier. This value allows uncommitted changes to be read. When a job inserts, updates, or deletes rows in a table, the changes are final after the job commits the transaction. If the job terminates and rolls back the transaction, the changes are removed and the table is restored to its previous state.

**Read committed**
Specifies that the connector prefixes the SELECT statement with a LOCK ROW FOR READ modifier and immediately commits each lookup. This value is valid only when the connector performs a lookup. This value prevents uncommitted changes from being read, but allows other transactions to modify rows that were read. The connector immediately commits each lookup to release the lock on any rows that were read, so that other transactions can update the row later while the lookup job is still in progress.

**Repeatable read**
Specifies that the connector prefixes the SELECT statement with a LOCK ROW FOR READ modifier. This value prevents other transactions from modifying the data that was read, but allows reads of phantom rows. The connector holds a read lock on any row that is read until the current transaction is committed. Therefore, rows that are reread during the execution of the job return the same result. Other transactions cannot update or delete rows that were read, but they can insert new rows (phantom rows) that the lookup can subsequently read.

**Serializable**
Specifies that the connector prefixes the SELECT statement with a LOCK table FOR READ modifier. The modifier references each table that is included in the SELECT statement. This value prevents other transactions from modifying the data that was read and prevents reads of phantom rows. The connector uses a table-level read lock until the end of the job. Reads are repeatable. While a lookup job is in progress, other transactions are not allowed to insert, update, or delete rows. If you select this value, you must construct the SELECT statement by using the SQL Builder window.
**Key column**

Use this property to specify the name of the column to use as the sort key.

If the connector uses multiple input links and you choose **Ordered** in the **Record ordering** field, use the **Key column** field to specify the field to use as the sort key.

**Null order**

Use this property to specify where to place null values in the sort order.

If the connector uses multiple input links and you choose **Ordered** in the **Record ordering** field, use the **Null order** field to specify where to put null values in relation to non-null values. The choices are **Before** and **After**.

**Limit settings**

Use the properties in this group to specify information about the connection sessions when you fetch or write rows.

**Load type**

Use this property to specify the type of bulk load to perform.

The following values are available:

- **Load**
  Performs a parallel bulk insert of high-volume data to a table. This value provides the fastest way to load high-volume data, but it cannot load tables that have secondary indexes, and it cannot load views. This load type locks the table during the load and uses one of the utility slots.

- **Update**
  Performs a parallel bulk insert, update, or delete of high-volume data. This value loads high-volume data faster than the **Stream** option, loads tables that have non-unique secondary indexes, and loads views. This load type locks the table during the load and uses one of the utility slots.

- **Stream**
  Performs a parallel bulk insert, update, or delete of low-volume data. This value loads high-volume data slower than the **Update** option, but it can load tables that have unique or non-unique secondary indexes, and it can load views. This load type does not use one of the utility slots and allows concurrent access to the table by using row-hash locks.

**Logging**

Use the properties in this group to specify information about logging. The properties in this group are used to log the values that are in each column when a SQL statement fails to insert, update, or delete a row. Each node that fails to insert, update, or delete rows prints the first row that failed on that node.

The properties in the **Logging** group are not available when **Write mode** property is set to **Bulk load**.

The following properties are included in the **Logging** group:

- Log column values on first row error
- Log key values only
- Column delimiter
Log column values on first row error
Use this property to specify whether the connector logs column values for the first row that failed on each node.

The Log column values on first row error property is not available when Write mode property is set to Bulk load.

The valid values are Yes or No. The default value is No.

Log key values only
Use this property to specify whether the connector logs the values of all columns or only key columns.

The Log key values only property is available when you specify Yes for the Log column values on first row error property. The Log key values only property is not available when Write mode property is set to Bulk load.

The valid values are Yes or No. The default value is No.

Log table
Use this property to specify the name of the restart log table for the bulk load.

Specify a value that is unique to the job instance and usage. If you leave this property blank, the default name is table_name_LOG. Double quotation marks are required if the table name contains special characters, as defined by Teradata. To put the log table in a different database than the source or target table, qualify the table name with a database name.

Logon mechanism
Use this property to set how the user logs in based on the user authentication.

A Logon mechanism is a security mechanism that authenticates the user. You can set the value of Logon mechanism property to TD2 or LDAP. The logon authentication for TD2 mechanism is handled by Teradata whereas for LDAP it is handled by Active directory. TD2 is the default Teradata Logon mechanism. The LDAP mechanism is used to support external authentication and authorization of users through the LDAP compliant directory.

The LDAP value is enabled only if the variant is set to 12.

Macro database
Use this property to specify the name of the database that contains any macros that are created or used by a stream load.

If the Server, User name, and Password properties are set, click the Database button to select a database on the Teradata server.

If you leave this property blank, the stream load uses the Teradata default.

Make duplicate copies
Use this property to specify whether duplicate copy protection is enabled for the table that is created.
The following values are available:

**Default**
Uses the default value for the Teradata database. The connector generates the CREATE TABLE statement without a FALLBACK or NO FALLBACK clause.

**Yes**
Specifies that duplicate copies of rows in the table are created and stored. The connector generates the CREATE TABLE statement with a FALLBACK clause.

**No**
Specifies that no duplicate copies of rows in the table are created. The connector generates the CREATE TABLE statement with a NO FALLBACK clause.

**Max buffer size**
Use this property to specify the maximum size of the request and response buffers for each connection session.

The default value of 0 specifies that the connector can set the maximum buffer size. Keep this setting under most circumstances. If you install a new version of the Teradata client that reduces the maximum buffer size, you can change the default setting to a value that the new Teradata client supports. Otherwise, the Teradata client might return an error that states that the request size exceeds the maximum.

If the **Server**, **User name**, and **Password** properties are set, click the **Max buffer size** button to select the maximum request or response size on the Teradata server.

**Max partition sessions**
Use this property to specify the maximum number of connection sessions per partition when you fetch or write rows.

This property is available if the **Access method** property is set to **Bulk**.

If the connector is in a source context, this property specifies the maximum number of sessions that the connector uses to fetch rows that are returned by the SELECT statement. If the connector is in a target context, this property specifies the maximum number of sessions that the connector uses to write rows for the INSERT, UPDATE, or DELETE statements.

The default value of 0 specifies that the connector chooses the maximum number of sessions per partition.

If the **Server**, **User name**, and **Password** properties are set, click the **AMPs** button to select the number of Access Module Processors (AMPs) on the Teradata server.

**Max sessions**
Use this property to specify the maximum number of connection sessions when you fetch or write rows.

This property is available if the **Access method** property is set to **Bulk**.

If the connector is in a source context, this property specifies the maximum number of sessions that the connector uses to fetch rows that are returned by the SELECT statement. If the connector is in a target context, this property specifies the
maximum number of sessions that the connector uses to write rows for the
INSERT, UPDATE, or DELETE statements.

The default value of 0 specifies that the connector chooses the maximum number
of sessions up to the number of available Access Module Processors (AMPs) on the
Teradata server.

If the Server, User name, and Password properties are set, click the AMPs button to
select the number of Access Module Processors (AMPs) on the Teradata server.

**Maximum bytes per character**

Use this property to specify the maximum number of bytes per characters that is
consumed by the specified parameters.

UTF8 contains 3 bytes per character, UTF16 contains 2 bytes per character, and
ASCII characters are represented by themselves as single bytes.

**Min sessions**

Use this property to specify the minimum number of connection sessions to use
when you fetch or write rows.

This property is available if the **Access method** property is set to Bulk.

If the connector is in a source context, this property specifies the minimum number
of sessions that the connector must use to fetch rows that are returned by the
SELECT statement. If the connector is in a target context, this property specifies the
minimum number of sessions that the connector must use to write rows for the
INSERT, UPDATE, or DELETE statements.

The job terminates if the connector is unable to establish the required number of
connection sessions. The default value of 0 specifies that there is no minimum.

If the Server, User name, and Password properties are set, click the AMPs button to
select the number of Access Module Processors (AMPs) on the Teradata server.

**Missing delete rows**

Use this property to specify whether to reject or ignore missing rows if a DELETE
statement fails to find a matching row.

This property is available if the **Write mode** property is set to Delete, Delete then
insert, or User-defined SQL.

The following values are available:

- **Default**
  Specifies that the Teradata Parallel Transporter Direct API decides the
default action. In most cases, the action is to reject the row.

- **Reject**
  Sends duplicate rows to one of the error tables.

- **Ignore**
  Ignores duplicate rows. No entry to an error table is created.

**Missing update rows**

Use this property to specify whether to reject or ignore missing rows if an
UPDATE statement fails to find a matching row.
This property is available if the **Write mode** property is set to *Update*, *Update then insert*, or *User-defined SQL*.

The following values are available:

**Default**
- Specifies that the Teradata Parallel Transporter Direct API decides the default action. In most cases, the action is to reject the row. However, if the **Write mode** property is set to *Update then insert*, the default action is to ignore the missing row and to perform the INSERT statement.

**Reject**
- Sends duplicate rows to one of the error tables.

**Ignore**
- Ignores duplicate rows. No entry to an error table is created.

**NLS Map Name**

**Format**

```
NLS\mapname
```

The NLS\mapname code converts data from the internal format to the external format and vice versa by using the specified map. \mapname is either a valid map name or one of the following: LPTR, CRT, AUX, or OS.

The **STATUS** function returns one of the following:

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Conversion successful</td>
</tr>
<tr>
<td>1</td>
<td>mapname invalid, string returned empty</td>
</tr>
<tr>
<td>2</td>
<td>Conversion invalid</td>
</tr>
<tr>
<td>3</td>
<td>Data converted, but result might be invalid (map could not deal with some characters)</td>
</tr>
</tbody>
</table>

**Pack size**

Use this property to specify the number of statements that a stream load puts into a multiple statement request.

The default value of 0 specifies that the stream load uses the Teradata default value.

**Parallel synchronization**

Use this property to indicate whether to synchronize parallel reads or writes by using a synchronization table.

The following values are available:

**Yes**
- Synchronizes parallel reads or writes.

**No**
- Runs in sequential mode.

**Password**

Use this property to specify the password that is associated with the user ID for the connection.

**Primary index type**

Use this property to specify whether the primary index for the key columns must be unique.
If none of the columns in the input schema have the Key attribute checked on the Columns tab, the connector specifies a primary index on the first column. If there are columns with the Key attribute checked, the key columns must be defined in the proper key order.

The following values are available:

**Non-unique**
- Specifies that the connector generates the CREATE TABLE statement with a PRIMARY INDEX clause without a UNIQUE qualifier. This value is the default.

**Unique**
- Specifies that the connector generates the CREATE TABLE statement with a UNIQUE PRIMARY INDEX clause.

**Progress interval**
- Use this property to specify the number of rows per partition to process before the connector displays a progress message.

By default, the connector displays a progress message for every 100,000 rows it processes in each partition. To change the interval, specify a new number of rows per partition. To disable progress messages, specify 0 for this property.

**Queryband expression**
- Use this property to specify the queryband expression for the `queryband` property.

A query band is a set of user-defined parameters that can be set on a session, a transaction, or both to identify the originating source of a query. Once a query band is defined, the system passes it to the Teradata database as a list of name:value pairs in a single quoted string. For example, 'ProjectName=dstage1'

**Record count**
- Use this property to specify the number of records to process before the connector commits the current transaction.

You must specify a number that is a multiple of the `Array size` property. The default value of 0 specifies that the transaction must be committed at the end of the job.

If the connector is in a target context and the `Access method` property is set to `Bulk`, the `Record count` property specifies how many records to process between checkpoints. The default value of 0 specifies that no checkpoints are done.

**Record ordering**
- Use this property to specify how to process records across multiple links.

Specify how to process records from multiple links. Choose one of the following:

- **All records** - All records from the first link are processed; then all records from the second link are processed; and so on.
- **First record** - One record from each link is processed until all records from all links have been processed.
- **Ordered** - Records are selected from the input links based on the order that you specify by using the `Key column`, `Sort order`, and `Null order` fields.
**Request type**

Use this property to specify whether to separate user-defined SQL statements into individual requests or to use a multi-statement request.

This property is available if the Access method property is set to Immediate. This property applies to the user-defined SQL statements in the Statements property or the File property.

The following values are available:

**Individual**

Separates the SQL statements into individual requests. The failure of any one statement does not affect the other statements.

**Multi-statement**

Combinesthe SQL statements intoa multi-statement request. The Teradata server runsmulti-statement requests atomically, so all statements fail if any one statement fails. Select this value if you want to update multiple tables, but you do not want a partial update if one of the updates fails. This value does not support arrays.

**Robust**

Use this property to specify whether a stream load uses robust restart logic.

If you set this property to Yes, the stream load uses robust restart logic. If you set this property to No, the stream load uses simpler restart logic.

**Schema reconciliation**

Use the properties in this group to specify how to reconcile differences between the design schema and the schema that is referenced or returned by the SQL statements.

**Select statement**

Use this property to define a SELECT statement to read rows from the database.

This property is required when the Generate SQL property is set to No.

You can use this property to invoke a Teradata macro or stored procedure.

To define a SELECT statement:

- Type a statement in the cell next to the Select statement field.
- Click the browse button to open the Select Statement window, which contains a large text box where you can type a statement.
- Click the Job Parameter button to specify a job parameter.
- Click the Tools button, then select Build New SQL, to open a graphical interface that helps you build, generate and validate statements.

**Serialize**

Use this property to specify whether a stream load runs multiple statements serially for each row.
The default value of Yes specifies that if there are multiple statements, the first statement must be run before the second statement, and each subsequent statement must be run in order. Use the default value if there is a dependency in the order in which statements must run.

**Server**

Use this property to specify the name of the Teradata Director Program ID (tdpid).

If you leave this property blank, the connector uses the value in the dbcname field of the System Parameter Block (clispb.dat file).

If the Server, User name, and Password properties are set, click the Server button to see version information about the Teradata server.

**Server character set**

Use this property to specify the character set for any CHAR and VARCHAR columns in the table that is created.

If you leave this property blank, the connector does not generate a CHARACTER SET clause for CHAR and VARCHAR columns. These columns get the default character set for the user in the User name property.

**Session**

Use the properties in this group to specify information about the session for read or write operations.

**Sleep**

Use this property to specify the number of minutes to wait before the connector retries a logon operation when the Teradata server is at the maximum number of concurrent bulk operations.

The default value of 0 specifies that the connector uses the Teradata default value.

**Sort order**

Use this property to specify whether to sort values in ascending or descending order.

If the connector uses multiple input links and you choose Ordered in the Record ordering field, use the Sort order field to specify whether to sort values in ascending or descending order.

**SQL**

Use the properties in this group to define the SQL statement that the connector uses to select, insert, update, or delete rows from the table.

**Start mode**

Use this property to specify whether to drop the error tables before the load begins or to restart a cancelled load.

The following modes are available:

- **Clean** Specifies that the connector drops the error tables, log table, and work table if they already exist before the load begins. This value is the default.
Restart
Species that the connector restarts a cancelled load. No action is performed for the Sync table action, Before SQL, Before SQL file, Before SQL (node), and Table action properties.

Start row
Use this property to specify the row number at which to begin the load operation.

The default value of 0 specifies that processing begins with the first row. Keep the default value if the connector runs in parallel mode. If you set this property to a value other than 0, the connector runs in sequential mode.

Statements
Use this property to specify the user-defined SQL statements to run for each input row.

This property is required when the User-defined SQL property is set to Statements.

Use semicolons to separate multiple SQL statements.

Stream load
Use the properties in this group to specify information about stream loads.

This property group is available if the Load type property is set to Stream.

Sync database
Use this property to specify the default database for parallel synchronization. This database is on the server where the synchronization table resides.

This property is available if the Parallel synchronization property is set to Yes.

If you leave this property blank, the connector uses the default database for the user in the Sync user property.

If the Server, User name, and Password properties are set, click the Database button to select a database on the server.

Sync ID
Use this property to specify the value that the connector enters in the SyncID column of the synchronization table.

This property is available if the Parallel synchronization property is set to Yes.

Specify a value that is unique to the job instance and usage. If you leave this property blank, the synchronization table must not be shared with other Teradata connector jobs or stages.

Sync password
Use this property to specify the password that is associated with the Sync user property.

This property is available if the Parallel synchronization property is set to Yes.
If you leave this property blank, the connector uses the password in the **Password** property.

### Sync poll

Use this property to specify the number of seconds to wait before each player rechecks that the synchronization table contains the proper state before processing continues.

This property is available if the **Parallel synchronization** property is set to **Yes**.

The default value of 0 specifies that the connector uses its default wait time.

### Sync server

Use this property to specify the name of the Teradata server where the synchronization table resides.

This property is available if the **Parallel synchronization** property is set to **Yes**.

If you leave this property blank, the connector expects the synchronization table to reside on the server that is specified by the **Server** property.

If the **Server**, **User name**, and **Password** properties are set, you can click the **Server** button to see version information about the Teradata server.

### Sync table

Use this property to specify the name of the Teradata table that synchronizes parallel reads or writes.

This property is available if the **Parallel synchronization** property is set to **Yes**.

This property is required when the connector runs in parallel mode, and the **Access method** property is set to **Bulk**.

The name **terasync** is reserved for the Teradata Enterprise stage and cannot be used. Double quotation marks are required if the table name contains special characters, as defined by Teradata.

If the **Server**, **User name**, and **Password** properties are set, click the **Table** button to select a table on the Teradata server.

### Sync table action

Use this property to specify the action to perform on the synchronization table before any data is processed by the connector.

This property is available if the **Parallel synchronization** property is set to **Yes**.

The following actions are available:

- **Append**
  Specifies that the synchronization table must already exist and that it might be shared by other jobs or stages.

- **Create**
  Specifies that the connector creates the synchronization table if it does not already exist. Otherwise, the connector uses the existing synchronization
table. This value is the default. The user in the Sync user property must have the CREATE TABLE privilege if the table does not exist.

Replace
Specifies that the name of the synchronization table is unique to the job instance and usage. If the synchronization table already exists, the connector drops the existing table and creates a new table. The user in the Sync user property must have DROP TABLE and CREATE TABLE privileges.

Truncate
Specifies that the name of the synchronization table is unique to the job instance and usage and that the table must already exist. The connector deletes all rows from the table. The user in the Sync user property must have the DELETE privilege.

Sync table cleanup
Use this property to specify the action to perform on the synchronization table after all data is processed by the connector.

This property is available if the Parallel synchronization property is set to Yes.

The following actions are available:

Drop  Specifies that the connector drops the synchronization table after all data is processed. The user in the Sync user property must have the DROP TABLE privilege.

Keep  Specifies that the connector does not drop the synchronization table. This value is the default.

Sync timeout
Use this property to specify the maximum number of seconds to retry an update to the synchronization table.

This property is available if the Parallel synchronization property is set to Yes.

If the timeout expires, the operation stops. The default value of 0 specifies that the connector uses its default timeout.

Sync user
Use this property to specify the name of the user ID to connect to the server where the synchronization table resides.

This property is available if the Parallel synchronization property is set to Yes.

If you leave this property blank, the connector uses the name in the User name property to connect to the server. The user must have privileges to SELECT, INSERT, and UPDATE the synchronization table.

If the Server, User name, and Password properties are set, click the User button to select a user name on the Teradata server.

Sync table write mode
Use this property to specify the mode to use when the connector writes to the synchronization table.
This property is available if the **Parallel synchronization** property is set to **Yes**, and the **Sync table action** property is set to **Append** or **Create**.

The following modes are available:

- **Insert**  Specifies that each job instance appends a new row to the synchronization table. This value is the default.

- **Delete then insert**  Specifies that each job instance deletes all rows in the synchronization table that match the value in the **Sync ID** property. Each job instance appends a new row to the synchronization table. The user in the **Sync user** property must have the **DELETE** privilege.

**Table action**

Use this property to specify the action to perform on the table before rows are inserted.

This property is required when the **Write mode** property is set to **Insert** or **User-defined SQL**.

The following actions are available:

- **Append**  Specifies that the table must already exist. This value is the default.

- **Create**  Specifies that the connector creates the table. If the table already exists, the connector raises an unrecoverable error and stops the job. The user in the **User name** property must have the **CREATE TABLE** privilege.

- **Replace**  Specifies that if the table already exists, the connector drops the existing table and creates a new table. The user in the **User name** property must have the **DROP TABLE** and **CREATE TABLE** privileges.

- **Truncate**  Specifies that the table must already exist. The connector deletes all rows from the table. The user in the **User name** property must have the **DELETE** privilege.

**Table free space**

Use this property to specify whether to generate the CREATE TABLE statement with or without a FREESPACE clause.

The following values are available:

- **Default**  Uses the default free space for the Teradata database. The connector generates the CREATE TABLE statement without a FREESPACE clause.

- **Yes**  Specifies that the connector generates the CREATE TABLE statement with a FREESPACE clause.

**Table name**

Use this property to specify the name of the table for the SQL statement that the connector generates.
In a source or lookup context, this property is required when the **Generate SQL** property is set to Yes. The connector uses the table name in the SELECT statement.

In a target context, this property is required when the **Generate SQL** property is set to Yes or when the **Table action** property is set to Create, Replace, or Truncate. The connector uses the table name in the INSERT, UPDATE, DELETE, CREATE TABLE, or DROP TABLE statement.

Double quotation marks are required if the table name contains special characters, as defined by Teradata. To specify the double quotation marks, use the **Enable quoted identifiers** property to automatically add double quotation marks to the table name.

In some job configurations, the **Table name** property is disabled and you cannot provide a table name value. If you either generate a data definition language (DDL) SQL statement by clicking Generate, or generate the data manipulation language (DML) SQL statement by selecting **Tools > Generate SQL statement**, when the **Table name** property is disabled, then the generated SQL statement uses the placeholder `<TABLENAME>` instead of the table name. You can then replace `<TABLENAME>` with the actual table name.

**Tenacity**

Use this property to specify the maximum number of hours that the connector continues to try to log on when the Teradata server is at the maximum number of concurrent bulk operations.

The default value of 0 specifies that the connector uses the Teradata default value.

**TMSM event options**

Use this property to provide access to the sub-properties that define options for the Teradata Multi-System Manager (TMSM) events.

When this property is set to Yes, the connector reports progress of the job by sending events and alerts to a TMSM server. TMSM event options are not available when the connector variant is 8.1. The following TMSM event options can be set through the connector properties: TMSM_UOWID, TMSM_UOWCLASS, and TMSM_UOWSOURCE.

If you set **TMSM event options** property to Yes, the TMSM sub-properties are available for you to use. If you set **TMSM event options** property to No, the sub-properties are not available to you. The default value is No.

**Transaction**

Use the properties in this group to tune parameters for transactional processing.

**Transaction mode**

Use this property to specify the type of transaction semantics.

This property does not affect bulk operations, but it does affect other SQL statements, such as those in the **Before SQL** and **After SQL** properties. This property also affects immediate reads and writes.

The following modes are available:
ANSI  This option is the default. If an error occurs, it does not affect the current transaction.

Teradata  Teradata automatically rolls back the current transaction if an error occurs.

**Truncate statement**

Use this property to define a DELETE statement to truncate the target table. Truncation is useful if you need to clear a table before you insert new data.

This property is required when the *Generate truncate statement at runtime* property is set to No.

To define a DELETE statement:

- Type a statement in the cell next to the **Truncate statement** property.
- Click the browse button to open the Truncate Statement window, which contains a large text box where you can type a statement.
- Click the **Job Parameter** button to use a job parameter.
- Click the **Generate** button to generate the statement automatically if you specified a value for the **Table name** property and defined column metadata on the Columns tab. You can modify the statement if necessary.

**Unused field action**

Use this property to specify whether to silently drop input fields that are not referenced in the SQL statements or to cancel the job.

The following actions are available:

- **Abort**  Specifies that the connector issues an unrecoverable error and cancels the job if the design schema contains any fields that are not referenced in the SQL statements. This value is the default.
- **Warn**  Specifies that the connector issues a warning for each unused input field that is dropped from the schema. The **Table action** property does not create columns for the unused fields that are dropped.
- **Drop**  Specifies that the connector silently drops all unused input fields. The unused fields are ignored, so they can have data types that are not supported by the connector. For rows that are sent to the reject link, the value of any dropped fields is set to NULL. The **Table action** property does not create columns for the unused fields that are dropped.
- **Keep**  Specifies that the connector keeps any unused input fields so that their values can be included in any rows that are sent to the reject link. In this case, the unused fields must have data types that are supported by the connector. The **Table action** property creates columns for the unused fields that are kept.

**UOW class**

Use this property to specify the classification of the unit of work. The connector will set this value for the TMSM_UOWCLASS environment variable.

**UOWID**

Use this property to specify a user-defined unit of work ID. The connector will set this value for the TMSM_UOWID environment variable.
This property is available when the Generate UOWID property is set to No.

**UOW source**

Use this property to specify the name of the system the data is sourced from. The connector will set this value for the TMSM_UOWSOURCE environment variable.

**Update load**

Use the properties in this group to specify information about update loads.

This property group is available if the Load type property is set to Update.

**Update statement**

Use this property to define an UPDATE statement to update rows in the database.

This property is required when the Generate SQL property is set to No and when the Write mode property is set to Update, Insert then update, or Update then insert.

To define an UPDATE statement:
- Type a statement in the cell next to the Update statement property.
- Click the browse button to open the Update Statement window, which contains a large text box where you can type a statement.
- Click the Job Parameter button to use a job parameter.
- Click the Build button to open the SQL Builder window, a graphical interface that helps you build the statement.

**User-defined SQL**

Use this property to identify the source of the user-defined SQL statements to run for each input row.

This property is required when the Write mode property is set to User-defined SQL.

You can use this property to invoke a Teradata macro or stored procedure. The call to the macro or stored procedure must be specified by the Statements property or be included in the file in the File property.

The following sources are available:

**Statements**

Specifies that the Statements property contains the SQL statements. This value is the default.

**File**

Specifies that the File property contains the name of a text file on the conductor node that contains the SQL statements. This value causes the connector to run in sequential mode.

**User name**

Use this property to specify the name of the user ID for connections to the Teradata server.

The user must have sufficient privileges to access the database. Single sign-on is not supported.
If the Server, User name, and Password properties are set, switch to different credentials by clicking the User button to select a user name on the Teradata server.

**Work table**

Use this property to specify the name of the work table.

This property is available if the Load type property is set to Load or Update.

Specify a value that is unique to the job instance and usage. If you leave this property blank, the default name is table_name_WT. Double quotation marks are required if the table name contains special characters, as defined by Teradata. To put the work table in a different database than the source or target table, qualify the table name with a database name.

**Write mode**

Use this property to specify the mode to use when the connector writes to a table.

The following modes are available:

- **Insert** Inserts new rows in the table. This value is the default and is the only valid value if the Load type property is set to Load.

- **Update** Updates matching rows in the table. Any rows in the input data that do not match rows in the table are ignored, but they can be rejected.

- **Delete** Deletes matching rows in the table. Any rows in the input data that do not match rows in the table are ignored, but they can be rejected.

- **Insert then update** Inserts new rows in the table, or if an error occurs, updates matching rows in the table. Any rows in the input data that do not match rows in the table are ignored, but they can be rejected. The UPDATE statement is not run if the INSERT statement succeeds. Select this value if you expect the insert to fail occasionally because rows already exist. You must set the Array size property to 1 and the Access method property to Immediate.

  Do not use this value if the Transaction mode property is set to Teradata. If an error occurs on the insert, Teradata automatically rolls back the current transaction.

- **Update then insert** Updates matching rows in the table, or if matching rows are not found, inserts new rows in the table. The INSERT statement is not run if the UPDATE statement successfully updates matching rows.

- **Delete then insert** Deletes matching rows in the table and adds new rows.

- **User-defined SQL**

  For every input row, runs the SQL statements in the Statements property or the File property in the User-defined SQL property group.
Chapter 4. Teradata API stage

The Teradata API stage reads data to any IBM InfoSphere DataStage stage from a Teradata database and writes data from any InfoSphere DataStage stage to a Teradata database. It also provides native data browsing and metadata import from the Teradata database to InfoSphere DataStage.

When you use IBM InfoSphere DataStage to access Teradata databases, you can choose from a collection of connectivity options. For most new jobs, use the Teradata Connector stage, which offers better functionality and performance than the Teradata API stage.

If you have jobs that use the Teradata API stage and want to use the connector, use the Connector Migration Tool to migrate jobs to use the connector.

This chapter describes the following topics:

- Functionality of Teradata API Stages
- Connecting to a Teradata Data Source
- Defining Teradata Input Data
- Defining Teradata Output Data
- Data Types

Introduction to Teradata API

The Teradata API stage enables IBM InfoSphere DataStage to read and write data to and from Teradata databases. It provides access to a database for row-by-row operations using CLI (Call Interface). It is not efficient for bulk operations but works well for small-table access.

The Teradata API stage offers the following advantages:

- Processing speed
- Support for the Teradata client software for Microsoft Windows and UNIX
- Simplified configuration on UNIX platforms
- Advanced support for target table DDL (Create and Drop Table)
- Native metadata import

This stage uses the Teradata CLI programming API for network-attached systems to let you connect and process SQL statements in the native Teradata environment.

In summary, the Teradata API stage lets you do the following for a target database:

- Read and write data
- Create and drop tables
- Import table and column definitions
- Browse native data with the custom GUI

Each Teradata API stage is a passive stage that can have any number of the following links:
Input links. Specify the data you are writing, which is a stream of rows to be loaded into a Teradata database. You can specify the data on an input link using an SQL statement generated by InfoSphere DataStage or constructed by you.

Output links. Specify the data you are extracting, which is a stream of rows to be read from a Teradata database. You can specify the data on an output link using an SQL SELECT statement generated by InfoSphere DataStage or constructed by you.

Reference output links. Each link represents rows that are key read from a Teradata database (using the key columns in a WHERE clause of the SELECT statement that is constructed by InfoSphere DataStage or by you.

You can also access Teradata databases using Stored Procedure (STP) stages. Use STP to include Teradata stored procedures as part of your job. For more information about this stage, see IBM InfoSphere DataStage and QualityStage Connectivity Guide for Stored Procedures.

Input Links
Input links specify the data you are writing, which is a stream of rows to be loaded into a Teradata database. You can specify the data on an input link by using an SQL statement generated by IBM InfoSphere DataStage or constructed by the user.

Output Links
Output links specify the data you are extracting, which is a stream of rows to be read from a Teradata database. You can specify the data on an output link by using an SQL SELECT statement generated by IBM InfoSphere DataStage or constructed by the user.

Reference Output Links
Each link represents rows that are key read from a Teradata database (using the key columns in a WHERE clause of the SELECT statement that is constructed by IBM InfoSphere DataStage or specified by the user).

Installing Teradata API
Before installing the stage, consult Teradata documentation for any specific configuration requirements.

Note: Very slow performance can occur as data is written if you run the Teradata server software on the same Microsoft Windows machine where you run Teradata API or ODBC (to Teradata).

To improve the performance, use two different machines for the Teradata client and Teradata server. This balances the load since the IBM InfoSphere DataStage server represents the Teradata client and the Teradata RDBMS represents the server.

Configuring the Environment for Teradata Stages
About this task
Some Teradata stages require the setting of environment variables in order to work correctly on a UNIX platform. To add or change an environment variable, include
any environment variables in the \textit{dsenv} file. The following table identifies the specific environment variables that are required.

\begin{table}[h]
\centering
\begin{tabular}{|l|l|}
\hline
For... & Set the following environment variable on the IBM InfoSphere DataStage server machine... \\
\hline
Teradata API stage & LD\_LIBRARY\_PATH \\
 & \textsc{PATH} \\
Teradata Load stage & LD\_LIBRARY\_PATH \\
 & \textsc{PATH} \\
Teradata MultiLoad/TPump/FastExport stage & LD\_LIBRARY\_PATH \\
 & \textsc{PATH} \\
\hline
\end{tabular}
\end{table}

The name of one particular environment variable, referred to as LD\_LIBRARY\_PATH above, differs depending on the platform. See the following to determine the correct name to use for your environment.

- If the platform is IBM AIX, use LIBPATH.
- If the platform is HP\_UX, use SHLIB\_PATH.
- If the platform is Linux, Solaris, or Tru64, use LD\_LIBRARY\_PATH.

\section*{Teradata Stages and the Parallel Canvas}

Some connectivity stages were designed for server jobs but can run on the parallel canvas. The default for all stages is Sequential. "In Parallel" mean you can set it to run in parallel, but this is \textit{not} the default. Use the following table to determine which connectivity server stages are available on the parallel canvas.

\begin{table}[h]
\centering
\begin{tabular}{|l|l|l|l|}
\hline
Connectivity Stage & Available on the Parallel Canvas in Windows & Available on the Parallel Canvas in UNIX & \begin{tabular}{c}Used as a Source or a Target or for Processing \end{tabular} & \begin{tabular}{c}Runs Sequentially or In Parallel \end{tabular} \\
\hline
Teradata API & Yes & Source or Target & Source: Sequential; Target: In Parallel \\
Teradata Load & Yes & Target & Sequential \\
Teradata MultiLoad/TPump/FastExport & Yes & Source or Target & Sequential \\
Teradata connector & Yes & Yes & Sequential and Parallel \\
\hline
\end{tabular}
\end{table}

\section*{Mapping String Data}

The purpose of the NONE map on the server canvas is to turn off mapping of string data in any stage in which the map is set, i.e., to pass the data through verbatim. This feature is handled differently on the parallel canvas. When you define string data (char, varchar, etc.), there is an additional field in the stage's
Columns grid called Extended. This can be set to blank or Unicode. If this option is set to blank, no mapping occurs (i.e. "NONE"); the map specified on the NLS tab is ignored. If this option is set to Unicode, the NLS map is applied. In order to read or write Japanese data, for example, set Extended to Unicode. When the job compiler detects this combination (char, varchar, etc. and Unicode), it generates the appropriate runtime code.

**Functionality of Teradata API Stages**

The Teradata API stage has the following functionality:

- Stream input, stream output, and reference output links.
- The ability to import table and column definitions from the target Teradata database and store them in the repository.
- NLS (National Language Support).
- Reject row handling.
- File names to contain your SQL statements.
- Data browsing, which is the ability to use the custom GUI for the stage to view sample native table data residing on the target Teradata database.

The following functionality is not supported:

- Bulk loading of Teradata tables. Use the Teradata Load stage for bulk loading into a Teradata database. For more information, see Chapter 6, “Teradata Load stage,” on page 97.
- Replacing the ODBC stage. The Teradata API stage does not replace the ODBC stage. Users who created jobs using the ODBC stage to access a Teradata database may continue to run these jobs.
- Stored procedures.
- Non-ANSI SQL statements in stage-generated SQL statements.
- Version-specific SQL statements in stage-generated SQL statements.
- Text and byte data types.

**Defining the Teradata Stage**

Using the GUI is easier than using grids to edit the values they contain. When you use the GUI to edit a Teradata API stage, the Teradata Stage dialog box opens. This dialog box has the Stage, Input, and Output pages (depending on whether there are inputs to and outputs from the stage):

- **Stage.** This page displays the name of the stage you are editing. The General tab defines the Teradata data source and login information. You can enter text to describe the purpose of the stage in the Description field. The properties on this tab define the connection to the Teradata data source. For details, see “Connecting to a Teradata data source” on page 67.

  The NLS tab defines a character set map to use with the stage. This tab appears only if you have installed NLS for IBM InfoSphere DataStage. For details, see “Defining Character Set Mapping” on page 67.

  **Note:** You cannot change the name of the stage from this dialog box.

- **Input.** This page is displayed only if you have an input link to this stage. It specifies the SQL table to use and the associated column definitions for each data input link. It also specifies how data is written and contains the SQL statement or call syntax used to write data to a Teradata table. It also specifies how to create the target table if desired and how to drop it if necessary.
Defining a Teradata API Stage

About this task

The main phases in defining a Teradata API stage from the Teradata Stage dialog box are as follows:

Procedure
1. Connect to a Teradata data source (see Connecting to a Teradata Data Source).
2. Optional. Define a character set map (see Defining Character Set Mapping).
3. Define the data on the input links (see Defining Teradata Input Data).
4. Define the data on the output links (see Defining Teradata Output Data).
5. Click OK to close this dialog box.

Connecting to a Teradata data source

The Teradata API connection parameters are set on the General tab of the Stage page.

The General tab contains the following fields and controls:

• Teradata Director. The Teradata Director Program ID (tdpid) that is associated with a particular Teradata server. The Teradata server has a unique tdpid. (See your system administrator for the identifier associated with the Teradata RDBMS that you plan to use.) If no value is given, the value in the dbname field in the clispb.dat file is used.
• User ID. The name to use to connect to the Teradata server. This user must have sufficient privileges to access the specified database and source and target tables.
• Password. The password associated with the specified user name. For security, it displays asterisks instead of the value you enter.
• Account ID. Your individual user account that is associated with User ID.
• Client character set. The Teradata client character set to use when connecting to the Teradata server. The default value is Default.
• Description. Optionally, describe the purpose of the Teradata API stage.

Defining Character Set Mapping

About this task

You can define a character set map for a stage. Do this from the NLS tab that appears on the Stage page. The NLS tab appears only if you have installed NLS.

Enter information for the following button and fields, if appropriate:

• Map name to use with stage. The default character set map is defined for the project or the job. You can change the map by selecting a map name from the list.
• **Use Job Parameter...** Specifies parameter values for the job. Use the format 
  #Param#, where Param is the name of the job parameter. The string #Param# is
  replaced by the job parameter when the job is run.

• **Show all maps**. Lists all the maps that are shipped with InfoSphere DataStage.

• **Loaded maps only**. Lists only the maps that are currently loaded.

---

### Defining Teradata Input Data

When you write data to a table in a Teradata database, the Teradata API stage has
an input link. Define the properties of this link and the column definitions of the
data on the Input page in the Teradata Stage dialog box.

### About the Input Page

The Input page has an **Input name** field, the **General**, **Columns**, and **SQL**
tabs, and the **Table Properties** (at the right of the **Drop table action** list box),

**Columns...**, and **View Data...** buttons.

- **Input name**. The name of the input link. Choose the link you want to edit from
  the **Input name** drop-down list box. This list displays all the input links to the
  Teradata API stage.

- Click the **Columns...** button to display a brief list of the columns designated on
  the input link. As you enter detailed metadata in the **Columns** tab, you can
  leave this list displayed.

- Click the **View Data...** button to start the Data Browser. This lets you look at the
  data associated with the input link.

### The General Tab

Use this tab to indicate how the SQL statements are created from an **Input** link on
the Teradata API stage.

The **General** tab contains the following fields:

**Query type**

This field contains **Use SQL Builder tool**, **Generate Update action from
Options and Columns Tab**, **Enter custom SQL statement**.

The **Use SQL Builder tool** specifies that the SQL statement is built using the
SQL Builder graphical interface. When this option is selected, the SQL Builder
button appears. When you click SQL Builder, the SQL Builder dialog box is
displayed. This causes the **Update action** property to appear.

**Generate Update action from Options and Columns tabs**. Causes the **Update
action** property to appear. Uses values from the **Options** and **Columns** tabs
and from **Update action** to generate the SQL.

The **Enter Custom SQL statement** writes the data using a user-defined SQL
statement, which overrides the default SQL statement generated by the stage. If
you choose this option, you enter the SQL statement on the SQL tab. See
“Using User-Defined SQL Statements” on page 72 for details.

**Update action**

Specifies which stage-generated SQL statements are used to update the target
table. Some update actions require key columns to update or delete rows. The
default is insert rows without clearing. Choose one of the following options:

- **Insert rows without clearing**, Inserts the new rows in the table.
Clear the table, then insert rows. Deletes the contents of the table before inserting the new rows. Transaction logging causes slower performance. When you click SQL Button, the Insert page opens.

Delete existing rows only. Deletes existing rows in the target file that have identical keys in the input rows. When you click SQL Button, the Delete page opens.

Replace existing rows completely. Deletes the existing rows, then adds the new rows to the table. When you click SQL Button, the Delete page opens. However, you must also complete an Insert page to accomplish the replace.

Update existing rows only. Updates the existing data rows. Any rows in the data that do not exist in the table are ignored. When you click SQL Button, the Update page opens.

Update existing or insert new rows. Updates the existing data rows before inserting new rows. Performance depends on the contents of the target table and the rows being processed in the job. If most rows exist in the target table, it is faster to update first. When you click SQL Button, the Update page opens. However, you must also complete an Insert page to accomplish the replace.

Insert new or update existing rows. Inserts the new rows before updating existing rows. Performance depends on the contents of the target table and the rows being processed in the job. If most rows do not exist in the target table, it is faster to insert first. When you click SQL Button, the Insert page opens. However you must also complete an Update page to accomplish the update.

Options tab

Use the Options tab to create or drop tables and to specify miscellaneous Teradata link options.

The Options tab contains the following fields:

- **Table name.** This field is editable when the update action is not User-defined SQL (otherwise, it is read-only). It is the name of the target table to update. You must specify Table name if you do not specify User-defined SQL. There is no default. You can also click the ... button at the right of the Table name field to browse the repository to select the table.

- **Transaction size.** The number of rows that the stage processes before committing a transaction to the database. The default is 100. This field applies only to databases that are set up for logging.

- **Create table action.** Choose one of the following options to create the target table in the specified database:
  - Do not create target table. Specifies that the target table is not created, and disables the Drop table action field and the Table Properties button (at the right of the field).
  - Generate DDL. Specifies that the stage generates the CREATE TABLE statement by using information obtained from the “Target Table” property, the column definitions grid, and the advanced table properties (see the Table Properties button later in this section).
  - User-defined DDL. Specifies that you enter the appropriate CREATE TABLE statement on the SQL tab.

- **Drop table action.** Controls the dropping of the target table before it is created by the stage. If you choose not to create the target table, this field is disabled. The list box displays the same items as the Create table action list box except that they apply to the DROP TABLE statement.
- **Table Properties button.** Click the button at the right of the Drop table action list box to display the Create Table Properties dialog box. You can then specify the following advanced table properties from this dialog box:
  - **Table Duplicates.** Controls duplicate row control. Use one of the following values:
    - NONE. Duplicate rows are not allowed in Teradata mode and but are allowed in ANSI mode. This is the default.
    - SET. Duplicate rows are not allowed. This is compatible with Teradata RDBMS tables from prior releases.
    - MULTISET. Duplicate rows are allowed. This is compliant with the ANSI standard.
  - **Table Copy.** Specifies whether to choose duplicate copy protection for the table. Use one of the following values:
    - NONE. This option is established by a CREATE DATABASE statement for the database in which the table is to be created. Do not use this clause in the CREATE TABLE statement.
    - FALLBACK. Duplicate copies of rows in the table are created and stored.
    - NOFALLBACK. Duplicate copies of rows in the table are not created and stored.
  - **Table Freespace.** Sets the percent of free space that remains on a cylinder during loading or update operations. The default value of 0 means that this clause is not used in the CREATE TABLE statement. Enter a number from 0 to 75.
  - **Table Block Size.** Sets the value of the data block size attribute to the unit specified in bytes. The default value of 0 means this clause is not used in the CREATE TABLE statement. Enter a number from 6144 to 32256.
- **Time data type** If you select REAL (the default value), the stage defines time columns with a data type of REAL. Time values are encoded as (hour*10000 + minute*100 + second), where second can include a fractional value. If you select TIME(n), the stage defines time columns as TIME(n), where n is the Scale value, in the range 0 through 6, representing the fractional seconds precision. The Length value for the time column must be 8 if Scale equals 0, or it must be 9+ if Scale is greater than 0.
- **Timestamp data type.** If you select CHAR(19) (the default value), the stage defines timestamp columns as CHAR(19). If you select TIMESTAMP(n), the stage defines timestamp columns as TIMESTAMP(n), where n is the Scale value, in the range 0 through 6, representing the fractional seconds precision. The Length value for the timestamp column must be 19 if Scale equals 0, or it must be 20+ if Scale is greater than 0.
- **Missing Update Rows** When a row to be updated in the target table is not available it should be sent to the reject link. Select Reject to send the row to reject link or Ignore to ignore the row.

**The Columns Tab**
This tab contains the column definitions for the data written to the table or file. The Columns tab behaves the same way as the Columns tab in the ODBC stage.

**The SQL Tab**
This tab contains the Query, Before, After, Generated DDL, and User-defined DDL tabs. Use these tabs to display the stage-generated SQL statement and the SQL statement that you can enter.
• **Query.** This tab contains the Query Type.

**Query type**

This option contains **Use SQL Builder tool**, **Generate SELECT clause from column list; enter other clauses**, **Enter SQL statement**, **Load SQL from a file at run time**.

The **Use SQL Builder tool** specifies that the SQL statement is built using the SQL Builder graphical interface. When this option is selected, the SQL Builder button appears. When you click SQL Builder, the SQL Builder dialog box is displayed.

The **Generate SELECT clause from column list; enter other clauses** specifies that InfoSphere DataStage generates the SELECT clause based on the columns you select on the Columns tab. When this option is selected, the SQL Clauses button appears. If you click SQL Clauses, the SQL Clauses dialog box is displayed. Use this dialog box to refine the SQL statement.

The **Enter Custom SQL statement** specifies that a custom SQL statement is built using the SQL tab.

The **Load SQL from a file at run time** specifies that the data is extracted using the SQL query in the path name of the designated file that exists on the server. Enter the path name for this file instead of the text for the query. You can edit the SQL statements using this option.

• **Before.** Contains the SQL statements executed before the stage processes any job data rows. The Before is the first SQL statement to be executed, and you can specify whether the job continues or aborts after failing to execute a Before SQL statement. It does not affect the transaction grouping scheme. The commit/rollback is performed on a per-link basis.

If the property value begins with FILE=, the remaining text is interpreted as a path name, and the contents of the file supplies the property value.

• **After.** Contains the After SQL statement executed after the stage processes any job data rows. It is the last SQL statement to be executed, and you can specify whether the job continues or aborts after failing to execute an After SQL statement. It does not affect the transaction grouping scheme. The commit/rollback is performed on a per-link basis.

If the property value begins with FILE=, the remaining text is interpreted as a path name, and the contents of the file supplies the property value.

• **Generated DDL.** Select **Generate DDL** or **User-defined DDL** from the Create table action field on the General tab to enable this tab. The CREATE statement field displays the CREATE TABLE statement that is generated from the column metadata definitions and the information provided on the Create Table Properties dialog box. If you select an option other than **Do not drop target table** from the Drop table action list, the DROP statement field displays the generated DROP TABLE statement for dropping the target table.

• **User-defined DDL.** Select **User-defined DDL** from the Create table action or Drop table action field on the General tab to enable this tab. The generated DDL statement is displayed as a starting point to define a CREATE TABLE and a DROP TABLE statement.

The DROP statement field is disabled if User-defined DDL is not selected from the Drop table action field. If **Do not drop target** is selected, the DROP statement field is empty in the Generated DDL and User-defined DDL tabs.

**Note:** Once you modify the user-defined DDL statement from the original generated DDL statement, changes made to other table-related properties do not affect the user-defined DDL statement. If, for example, you add a new column in
the column grid after modifying the user-defined DDL statement, the new column appears in the generated DDL statement but does not appear in the user-defined DDL statement.

Writing Data to Teradata

The following sections describe the differences when you use stage-generated or user-defined SQL INSERT, DELETE, or UPDATE statements to write data from IBM InfoSphere DataStage to a Teradata database.

Using Generated SQL Statements

About this task

By default, IBM InfoSphere DataStage writes data to a Teradata table by using an SQL INSERT, DELETE, or UPDATE statement that it constructs. The generated SQL statement is automatically constructed by using the InfoSphere DataStage table and column definitions that you specify in the input properties for this stage. The Generated tab on the SQL tab displays the SQL statement used to write the data.

To use a generated statement:

Procedure

1. Enter a table name in the Table name field on the General tab of the Input page.
2. Specify how you want the data to be written by choosing an option from the Update action drop-down list box. See “The General Tab” for a description of the update actions.
3. Optional. Enter a description of the input link in the Description field.
5. Edit the Columns grid to specify column definitions for the columns you want to write.

The SQL statement is automatically constructed using your chosen update action and the columns you have specified. You can now optionally view this SQL statement.

6. Click the SQL tab on the Input page, then the Generated tab to view this SQL statement. You cannot edit the statement here, but you can always access this tab to select and copy parts of the generated statement to paste into the user-defined SQL statement.

7. Click OK to close this dialog box. Changes are saved when you save your job design.

Using User-Defined SQL Statements

About this task

Instead of writing data using an SQL statement constructed by the stage, you can enter your own SQL INSERT, DELETE, or UPDATE statement for each Teradata input link. Ensure that the SQL statement contains the table name, the type of update action you want to perform, and the columns you want to write.

To use your own SQL statement:
Procedure
1. Choose User-defined SQL from the Update action drop-down list box on the General tab of the Input page.
2. Click the SQL tab, then the User-defined tab. The User-defined tab page opens.
   By default you see the stage-generated SQL statement. You can edit this statement or enter your own SQL statement to write data to the target Teradata tables. This statement must contain the table name, the type of update action you want to perform, and the columns you want to write.
   If the property value begins with [FILE], the remaining text is interpreted as a path name, and the contents of the file supplies the property value.
   When writing data, the INSERT statements must contain a VALUES clause with a parameter marker ( ? ) for each stage input column. UPDATE statements must contain a SET clause with parameter markers for each stage input column. UPDATE and DELETE statements must contain a WHERE clause with parameter markers for the primary key columns. If you specify multiple SQL statements, each is executed as a separate transaction. Terminate individual SQL statements with a semicolon (;). Use a double semicolon (;;) to indicate the end of the command batch. You cannot combine multiple INSERT, UPDATE, and DELETE statements in one batch. You must execute each statement in a separate command batch.
   The parameter markers must be in the same order as the associated columns listed in the stage properties. For example:
   `INSERT emp (emp_no, emp_name) VALUES (?, ?)`
   The size of this box changes proportionately when the main window is resized to conveniently display very long or complex SQL statements.
   Unless you specify a user-defined SQL statement, the stage automatically generates an SQL statement.
3. Click OK to close this dialog box. Changes are saved when you save your job design.

Defining Teradata Output Data

When you read data from a Teradata data source, the Teradata API stage has an output link. The properties of this link and the column definitions of the data are defined on the Output page in the Teradata Stage dialog box.

About the Output Page

The Output page has an Output name field, the General, Columns, Selection, and SQL tabs, and the Columns... and View Data... buttons. The tabs displayed depend on how you specify the SQL statement to output the data.

- **Output name.** The name of the output link. Choose the link you want to edit from the Output name drop-down list box. This list displays all the output links.
- Click the Columns... button to display a brief list of the columns designated on the output link. As you enter detailed metadata in the Columns tab, you can leave this list displayed.
- Click the View Data... button to start the Data Browser. This lets you look at the data associated with the output link.
The General Tab

This tab is displayed by default. It provides the type of query and, where appropriate, a button to open an associated dialog box. The General tab contains the following field:

**Query Type**

This field contains **Use SQL Builder tool**, **Generate SELECT clause from column list; enter other clauses**, **Enter SQL statement**, and **Load SQL from a file at run time**.

The **Use SQL Builder tool** specifies that the SQL statement is built using the SQL Builder graphical interface. When this option is selected, the SQL Builder button appears. When you click SQL Builder, the SQL Builder dialog box is displayed.

The **Generate SELECT clause from column list; enter other clauses** specifies that InfoSphere DataStage generates the SELECT clause based on the columns you select on the Columns tab. When this option is selected, the SQL Clauses button appears. If you click SQL Clauses, the SQL Clauses dialog box is displayed. Use this dialog box to refine the SQL statement.

The **Enter Custom SQL statement** specifies that a custom SQL statement is built using the SQL tab.

The **Load SQL from a file at run time** specifies that the data is extracted using the SQL query in the path name of the designated file that exists on the server. Enter the path name for this file instead of the text for the query. You can edit the SQL statements using this option.

**Description**

Optionally enter a description of the output link.

The Options tab

Use this tab to specify the number of prefetch rows, characteristics of TIME and TIMESTAMP, and the isolation level.

The Options tab contains the following fields:

- **Prefetch rows.** The number of rows that Teradata returns when the stage fetches data from the source tables. Specifying a value greater than 1 improves performance (memory usage increases to accommodate buffering multiple rows).
- **Time data type.** If you select REAL (the default value), the stage expects time columns to be defined with a data type of REAL. If you select TIME(n), the stage expects time columns to be defined as TIME(n), where n is the Scale value, in the range 0 through 6, representing the fractional seconds precision. The Length value for the time column must be 8 if Scale equals 0, or it must be 9 + Scale.
- **Timestamp data type.** If you select CHAR(19) (the default value), the stage expects timestamp columns to be defined as CHAR(19). If you select TIMESTAMP(n), the stage expects timestamp columns to be defined as TIMESTAMP(n), where n is the Scale value, in the range 0 through 6, representing the fractional seconds precision. The Length value for the timestamp column must be 19 if Scale equals 0, or it must be 20 + Scale.
- **Isolation Level.** Sets the isolation level. The values are:
  - **None.** Uses the default isolation level for the database.
  - **Read Uncommitted.** Allows the reading of changes before they are committed (dirty reads). When a job updates a table (inserts, updates or deletes rows), those changes are not final until the job commits the transaction. If the job
 aborts and rolls back the transaction, then those changes are backed out, and
the table is restored to the state it was in when the transaction was initiated.
Selecting Read Uncommitted allows the stage to read uncommitted changes
while another transaction is updating the table.

- Read Committed. Prevents the reading of changes before they are committed
but allows other transactions to modify rows that have been read. Normally
when a job performs a lookup in a table, the rows that are read are locked
until the end of the job; thus if the job rereads the same row it will get the
same result. Selecting Read Committed causes the stage immediately to
release the lock on a row that has been read so that other transactions can
later update the row while the lookup is still in progress.

- Repeatable Read. Prevents any transactions from modifying any data that has
been read but allows the reading of phantom rows. Selecting Repeatable
Read causes the stage to hold a read lock on any row that is read until the
end of the job. This guarantees that rows that are reread during the execution
of the job will return the same result. Other transactions cannot update or
delete rows that have been read, but they can insert new rows that the lookup
can then subsequently read. These are known as phantom rows.

- Serializable. Prevents any transactions from modifying any data that has
been read and prevents the reading of phantom rows. Selecting Serializable
causes the stage to use a table-level read lock until the end of the job. Thus,
transactions cannot insert, update or delete rows while the table is being read.
Reads are repeatable, and it is not possible to read phantom rows. Other
transactions cannot update any rows in the table while the lookup is in
progress.

• Use column derivation fields. If not selected (the default), the Derivation field
on the Columns tab is ignored. If selected, any SQL expressions in the
Derivation field on the Columns tab appear in the select list of the generated
SQL SELECT statement. If an expression is not specified in the Derivation field,
the column name appears in the select list of the generated SQL SELECT
statement.

The Columns Tab
This tab contains the column definitions for the data being output on the chosen
link. The column definitions for reference links require a key field. Key fields join
reference inputs to a Transformer stage. Teradata API key reads the data by using a
WHERE clause in the SQL SELECT statement.

The Selection Tab
This tab is used primarily with generated SQL queries. It contains optional SQL
SELECT clauses for the conditional extraction of data. These clauses are appended
to the generated SQL statements.

The SQL Tab
Use this tab page to build the SQL statements used to read data from Teradata
databases. It contains the Query, Before, and After tab pages:

• Query. This tab contains the Query Type.

Query type
This option contains Use SQL Builder tool, Generate SELECT clause from
column list; enter other clauses, Enter SQL statement, Load SQL from a
file at run time.
The **Use SQL Builder tool** specifies that the SQL statement is built by using the SQL Builder graphical interface. When this option is selected, the SQL Builder button appears. When you click SQL Builder, the SQL Builder dialog box is displayed.

The **Generate SELECT clause from column list; enter other clauses** specifies that IBM InfoSphere DataStage generates the SELECT clause based on the columns you select on the Columns tab. When this option is selected, the SQL Clauses button appears. If you click SQL Clauses, the SQL Clauses dialog box is displayed. Use this dialog box to refine the SQL statement.

The **Enter Custom SQL statement** specifies that a custom SQL statement is built by using the SQL tab.

The **Load SQL from a file at run time** specifies that the data is extracted by using the SQL query in the path name of the designated file that exists on the server. Enter the path name for this file instead of the text for the query. You can edit the SQL statements by using this option.

- **Before.** Contains the SQL statements executed before the stage processes any job data rows. The Before is the first SQL statement to be executed, and you can specify whether the job continues or aborts after failing to execute a Before SQL statement. It does not affect the transaction grouping scheme. The commit/rollback is performed on a per-link basis.
  
  If the property value begins with FILE=, the remaining text is interpreted as a path name, and the contents of the file supplies the property value.

- **After.** Contains the After SQL statement executed after the stage processes any job data rows. It is the last SQL statement to be executed, and you can specify whether the job continues or aborts after failing to execute an After SQL statement. It does not affect the transaction grouping scheme. The commit/rollback is performed on a per-link basis.
  
  If the property value begins with FILE=, the remaining text is interpreted as a path name, and the contents of the file supplies the property value.

**Related reference:**

- ["SQL Clauses Dialog Box" on page 77](#)

**Reading Data from Teradata**

The following sections describe the differences when you use generated queries or user-defined queries to read data from a Teradata database into IBM InfoSphere DataStage.

**Using Generated Queries**

**About this task**

By default, the stage extracts data from a Teradata data source using an SQL SELECT statement that it constructs. The SQL statement is automatically constructed using the table and column definitions that you entered in the stage output properties.

When you select **Generate SELECT clause from column list; enter other clauses**, data is extracted from a Teradata database using an SQL SELECT statement constructed by the stage. Also the **SQL Clauses** button appears. Click **SQL Clauses**. The SQL Clauses dialog box appears. SQL SELECT statements have the following syntax:

SQL SELECT statements have the following syntax:
SELECT clause FROM clause
[WHERE clause]
[GROUP BY clause]
[HAVING clause]
[ORDER BY clause];

When you specify the tables to use and the columns to be output from the Teradata API stage, the SQL SELECT statement is automatically constructed and can be viewed by clicking the SQL tab on the Output page.

For example, if you extract the Name, Address, and Phone columns from a table called Table1, the SQL statement displayed on the SQL tab is:
SELECT Name, Address, Phone FROM Table1;

The SELECT and FROM clauses are the minimum required and are automatically generated by the Teradata API stage. However, you can use any of these SQL SELECT clauses:
- **SELECT clause.** Specifies the columns to select from the database.
- **FROM clause.** Specifies the tables containing the selected columns.
- **WHERE clause.** Specifies the criteria that rows must meet to be selected.
- **GROUP BY clause.** Groups rows to summarize results.
- **HAVING clause.** Specifies the criteria that grouped rows must meet to be selected.
- **ORDER BY clause.** Sorts selected rows.

**SQL Clauses Dialog Box:**
Use this dialog box to enter FROM, WHERE, or any other SQL clauses. It contains the Clauses and SQL tabs.
- **Clauses tab.** Use this tab to build column-generated SQL queries. It contains optional SQL clauses for the conditional extraction of data. The Clauses tab is divided into three panes.
  - **FROM clause (table name):** Allows you to name the table against which the SQL statement runs. To access Table Definitions, click ... (ellipses).
  - **WHERE clause.** Allows you to insert an SQL WHERE clause to specify criteria that the data must meet before being selected.
  - **Other clauses.** Allows you to insert a GROUP BY, HAVING, or ORDER BY clause to sort, summarize, and aggregate data.
- **SQL Tab.** Use this tab to display the SQL statements that read data from the database. You cannot edit these statements, but you can use Copy to copy them to the Clipboard for use elsewhere.

**Using Additional SQL Select Clauses**
**About this task**

If you want to use the additional SQL SELECT clauses, you must enter them on the Selection tab of the Output page. These clauses are appended to the SQL statement that is generated by the stage. If this link is a reference link, only the WHERE clause is enabled.

The Selection tab is divided into two areas (panes). You can resize an area by dragging the split bar for displaying long SQL clauses.
- **WHERE clause.** This text box allows you to insert an SQL WHERE clause to specify criteria that the data must meet before being selected.
• Other clauses. This text box allows you to insert a GROUP BY, HAVING, or ORDER BY clause.

Using User-Defined Queries
About this task

Instead of using the SQL statement constructed by the Teradata API stage, you can enter your own SQL statement for each Teradata output link.

Procedure
1. Select User-defined SQL query from the Query type drop-down list box on the General tab of the Output page. The User-defined tab on the SQL tab is enabled. It looks like the User-defined tab for the input link.
2. You can edit the statements or drag and drop the selected columns into your user-defined SQL statement. You must ensure that the table definitions for the output link are correct and represent the columns that are expected.
   If your entry begins with [FILE], the remaining text is interpreted as a path name, and the contents of the file supplies the text for the query.
3. Click OK to close this dialog box. Changes are saved when you save your job design.

SQL statements and the Teradata API stage

You can create SQL statements in the Teradata API stage from input and output links.

From an input link, you can create INSERT statements, UPDATE statements, and DELETE statements. From an output link, you can create SELECT statements.

You have four options for creating SQL statements:
• Using the SQL builder.
• Generating statements based on the values provided to the API stage.
• Entering user-defined SQL statements.
• Loading SQL statements from a file at run time.

Accessing the SQL builder from a server stage

You use the SQL builder to create SQL statements by using a graphical interface.

About this task

To access the SQL builder from a server stage:

Procedure
1. Select Use SQL Builder tool as the Query Type from the General tab of the input or output link or from the SQL tab.
2. Click the SQL Builder button. The SQL Builder window opens.
Data Types

The Teradata API stage assumes a Char or VarChar column is Unicode if the value specified in Display on the Columns tab is double the value specified in Length. Therefore, if you select the option to drop and recreate the table, the Generated DLL tab shows a CHARACTER SET UNICODE clause on those columns.

Mapping Data Types

You can map IBM InfoSphere DataStage data types to Teradata data types. When Create Table is set to Yes for input links, the target table is created by using the input link column definitions and the input link properties that define the properties for the target table.

Reject Row Handling

The Teradata API stage supports reject row handling. It does not buffer rows but sends them to Teradata as soon as it gets them. If the insert fails, the row is rejected.
Chapter 5. Teradata Enterprise stage

The Teradata Enterprise stage is a database stage that you can use to read data from and write data to a Teradata database.

When you use IBM InfoSphere DataStage to access Teradata databases, you can choose from a collection of connectivity options. For most new jobs, use the Teradata Connector stage, which offers better functionality and performance than the Teradata Enterprise stage.

If you have jobs that use the Teradata Enterprise stage and want to use the connector, use the Connector Migration Tool to migrate jobs to use the connector.

Overview

The Teradata Enterprise stage can have a single input link or a single output link. This is explained in the following diagrams.

When you edit a Teradata Enterprise stage, the Teradata Enterprise stage editor appears. This is based on the generic stage editor.

The stage editor has up to three pages, depending on whether you are reading or writing a file:

- **Stage Page** This is always present and is used to specify general information about the stage.
- **Input Page** This is present when you are writing to a Teradata database. This is where you specify details about the data being written.
- **Output Page** This is present when you are reading from a Teradata database. This is where you specify details about the data being read.
Accessing Teradata Databases

You must complete certain steps to access Teradata databases using the Teradata enterprise stage.

Installing the Teradata Tools and Utilities

About this task

You must install Teradata Tools and Utilities on all nodes that will run IBM InfoSphere DataStage parallel jobs. See the installation instructions supplied by Teradata. (You need system administrator status for the install.)

Creating Teradata User

About this task

You must set up a Teradata database user (this is the user that will be referred to by the DB options property in the Teradata stage). The user must be able to create tables and insert and delete data. The database for which you create this account requires at least 100 MB of PERM space and 10 MB of SPOOL. Larger allocations may be required if you run large and complex jobs. (You need database administrator status in order to create user and database.)

The example below shows you how to create the orchserver account. The user information is stored in the terasync table. The name of the database in this example is userspace. The following four commands for BTEQ set up the account:

```
CREATE USER orchserver FROM userspace AS
PASSWORD = orchserver
PERM = 100000000
SPOOL = 10000000
```

Once the account is set up, issue the following command:

```
GRANT select ON dbc TO orchserver;
```

Creating a Database Server

About this task

If you want to use a pre-existing Teradata user, you only need install a database server and configure it to use a new database. Install the new database server with the same PERM and SPOOL values as shown above. Here is an example of creating a database server called devserver using table userspace:

```
CREATE DATABASE devserver FROM userspace AS
PERM = 100000000
SPOOL = 10000000
GRANT create table, insert, delete, select ON devserver TO orchclient;
GRANT create table, insert, delete, select ON devserver TO orchserver;
```

Teradata Databases - Points to Note

NLS Support and Teradata Database Character Sets

The Teradata database supports a fixed number of character set types for each char or varchar column in a table. Use this query to get the character set for a Teradata column:

```
SELECT 'column name', chartype FROM dbc.columns
WHERE tablename = 'table_name'
```
The database character set types are:

- **Latin**: chartype=1. The character set for U.S. and European applications which limit character data to the ASCII or ISO 8859 Latin1 character sets. This is the default.
- **Unicode**: chartype=2. 16-bit Unicode characters from the ISO 10646 Level 1 character set. This setting supports all of the ICU multibyte character sets.
- **KANJISJIS**: chartype=3. For Japanese third-party tools that rely on the string length or physical space allocation of KANJISJIS.
- **Graphic**: chartype=4. Provided for IBM DB2 compatibility.

**Note**: The KANJI1: chartype=5 character set is available for Japanese applications that must remain compatible with previous releases; however, this character set will be removed in a subsequent release because it does not support the new string functions and will not support future characters sets. We recommend that you use the set of SQL translation functions provided to convert KANJI1 data to Unicode.

IBM InfoSphere DataStage maps characters between Teradata columns and the internal UTF-16 Unicode format by using the project default character set map unless this has been overridden at the job level (on the Job Properties dialog box) or the stage level (by using the NLS Map tab).

The file tera_cs.txt in the directory $APT_ORCHHOME/etc maps InfoSphere DataStage NLS character sets to Teradata character sets. For example, we select the EUC_JP as the NLS map for the current project. EUC_JP is the NLS character set for Japanese, 118 is the Teradata character set code for the KANJIEUC_0U character set. EUC_JPN is mapped to 118 in tera_cs.txt as follows:

<table>
<thead>
<tr>
<th>EUC_JP</th>
<th>118</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASC_JPN_EUC</td>
<td>118</td>
</tr>
<tr>
<td>SJIS</td>
<td>119</td>
</tr>
</tbody>
</table>

On reading, InfoSphere DataStage converts a Teradata varchar(n) field to ustring [n/min] where min is the minimum size in bytes of the largest codepoint for your specified character set. On writing, ustring data is converted to the specified character set and written to a char or varchar column in the Teradata database; the type is ustring[n*max] where max is the maximum size in of the largest codepoint for your specified character set.

InfoSphere DataStage also supports the use of Unicode character data in user names, passwords, column names, table names, and database names.

### Column Name and Data Type Conversion

IBM InfoSphere DataStage column names are case sensitive, Teradata column names are not. You must ensure that the InfoSphere DataStage column names are unique regardless of case.

Both InfoSphere DataStage and Teradata columns support nulls, and a InfoSphere DataStage column that contains a null is stored as a null in the corresponding Teradata column.

The Teradata stage automatically converts InfoSphere DataStage data types to Teradata data types and vice versa as shown in the following table:
### Table 9. Data type conversion in Teradata Enterprise stage

<table>
<thead>
<tr>
<th>InfoSphere DataStage SQL Data Type</th>
<th>Underlying Data Type</th>
<th>Teradata Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>date</td>
<td>date</td>
</tr>
<tr>
<td>Decimal</td>
<td>decimal (p, s)</td>
<td>numeric (p, s)</td>
</tr>
<tr>
<td>Double</td>
<td>dfloat</td>
<td>double precision</td>
</tr>
<tr>
<td>Double</td>
<td>dfloat</td>
<td>float</td>
</tr>
<tr>
<td>Double</td>
<td>dfloat</td>
<td>real</td>
</tr>
<tr>
<td>TinyInt</td>
<td>int8</td>
<td>byteint</td>
</tr>
<tr>
<td>SmallInt</td>
<td>int16</td>
<td>smallint</td>
</tr>
<tr>
<td>Integer</td>
<td>int32</td>
<td>integer</td>
</tr>
<tr>
<td>BigInt</td>
<td>int64</td>
<td>unsupported</td>
</tr>
<tr>
<td>LongVarBinary VarBinary</td>
<td>raw</td>
<td>varbyte (default)</td>
</tr>
<tr>
<td>Binary</td>
<td>raw [fixed_size]</td>
<td>byte (fixed_size)</td>
</tr>
<tr>
<td>LongVarBinary VarBinary</td>
<td>raw [max=size]</td>
<td>varbyte (size)</td>
</tr>
<tr>
<td>LongVarBinary VarBinary</td>
<td>raw [max=size]</td>
<td>graphic (c)</td>
</tr>
<tr>
<td>LongVarBinary VarBinary</td>
<td>raw [max=size]</td>
<td>vargraphic (size)</td>
</tr>
<tr>
<td>LongVarBinary VarBinary</td>
<td>raw [max=size]</td>
<td>long vargraphic</td>
</tr>
<tr>
<td>Float</td>
<td>sfloat</td>
<td>unsupported</td>
</tr>
<tr>
<td>Real</td>
<td></td>
<td>unsupported</td>
</tr>
<tr>
<td>LongVarChar VarChar</td>
<td>string</td>
<td>varchar (default length)</td>
</tr>
<tr>
<td>Unknown Char</td>
<td>string [fixed_size]</td>
<td>char (fixed_size)</td>
</tr>
<tr>
<td>LongVarChar VarChar</td>
<td>string [max=size]</td>
<td>varchar (size)</td>
</tr>
<tr>
<td>LongVarChar VarChar</td>
<td>string [max=size]</td>
<td>long varchar (size)</td>
</tr>
<tr>
<td>Time</td>
<td>time</td>
<td>unsupported</td>
</tr>
<tr>
<td>Timestamp</td>
<td>timestamp</td>
<td>unsupported</td>
</tr>
<tr>
<td>TinyInt</td>
<td>uint8</td>
<td>unsupported</td>
</tr>
<tr>
<td>SmallInt</td>
<td>uint16</td>
<td>unsupported</td>
</tr>
<tr>
<td>Integer</td>
<td>uint32</td>
<td>unsupported</td>
</tr>
</tbody>
</table>

InfoSphere DataStage columns are matched by name and data type to columns of the Teradata table, but they do not have to appear in the same order. The following rules determine which InfoSphere DataStage columns are written to a Teradata table:

- If there are InfoSphere DataStage columns for which there are no matching columns in the Teradata table, the job terminates. However, you can deal with
this by setting the **Silently drop columns not in table** property (see “Silently Drop Columns Not in Table” on page 90) or by dropping the column before you write the data.

- If the Teradata table contains a column that does not have a corresponding InfoSphere DataStage column, Teradata writes the column’s default value into the field. If no default value is defined for the Teradata column, Teradata writes a null. If the field is not nullable, an error is generated and the job fails.

### Restrictions and Limitations when Writing to a Teradata Database

There are the following limitations when using a Teradata Enterprise stage to write to a Teradata database:

- A Teradata row may contain a maximum of 256 columns.
- While the names of IBM InfoSphere DataStage columns can be of any length, the names of Teradata columns cannot exceed 30 characters. Rename your columns if necessary or specify the **Truncate column names** property to deal automatically with overlength column names (see “Truncate Column Names” on page 91).
- InfoSphere DataStage assumes that the stage writes to buffers whose maximum size is 32 KB. However, you can override this and enable the use of 64 KB buffers by setting the environment variable APT_TERA_64K_BUFFERS (see IBM InfoSphere DataStage and QualityStage Parallel Job Advanced Developer’s Guide).
- When writing to Teradata, the InfoSphere DataStage column definitions should not contain fields of the following types:
  - BigInt (int64)
  - Unsigned integer of any size
  - String, fixed- or variable-length, longer than 32 KB
  - Raw, fixed- or variable-length, longer than 32 KB
  - Subrecord
  - Tagged aggregate
  - Vectors
    - If InfoSphere DataStage tries to write data whose columns contain a data type listed above, the write is not begun and the job containing the stage fails. You can convert unsupported data types by using the Modify stage.
  - The Teradata Enterprise stage uses a distributed FastLoad to write the data and is subject to all the restrictions on FastLoad. Briefly, these are:
    - There is a limit to the number of concurrent FastLoad and FastExport jobs in Teradata.
    - Each instance of the Teradata stage using FastLoad or FastExport in a job counts towards this limit.

### Restrictions on Reading a Teradata Database

The Teradata Enterprise stage uses a distributed FastExport to access the data and is subject to all the restrictions on FastExport. Briefly, these are:

- There is a limit to the number of concurrent FastLoad and FastExport jobs.
- Each instance of the Teradata stage using FastLoad or FastExport in a job counts towards this limit.
- Aggregates and most arithmetic operators in the SELECT statement are not allowed.
- The use of the USING modifier is not allowed.
Must Do's

IBM InfoSphere DataStage has many defaults which means that it can be very easy to include Teradata Enterprise stages in a job. This section specifies the minimum steps to take to get a Teradata Enterprise stage functioning. InfoSphere DataStage provides a versatile user interface, and there are many shortcuts to achieving a particular end, this section describes the basic method, you will learn where the shortcuts are when you get familiar with the product.

The steps required depend on what you are using a Teradata Enterprise stage for.

Writing a Teradata Database

About this task
- In the input link Properties tab (see "Input Link Properties Tab" on page 88), under the Target category:
  - Specify the Table you are writing.
  - Specify the write mode (by default IBM InfoSphere DataStage appends to existing tables, you can also choose to create a new table, replace an existing table, or keep existing table details but replace all the rows).
  - Under the Connection category:
    - You can either manually specify a connection string, or have InfoSphere DataStage generate one for you by using a user name and password you supply. Either way you need to supply a valid user name and password. InfoSphere DataStage encrypts the password when you use the auto-generate option.
    - Specify the name of the server hosting Teradata.
  - Ensure column metadata has been specified for the write.

Reading a Teradata Database

About this task
- In the output link Properties tab (see "Output Link Properties Tab" on page 93), under the Source category:
  - Choose a Read Method. This is Table by default directly from a table, but you can also choose to read by using auto-generated SQL or user-generated SQL.
  - Specify the table to be read.
  - If using a Read Method of user-generated SQL, specify the SELECT SQL statement to use. IBM InfoSphere DataStage provides the auto-generated statement as a basis, which you can edit as required.
  - Under the Connection category:
    - You can either manually specify a connection string, or have InfoSphere DataStage generate one for you by using a user name and password you supply. Either way you need to supply a valid user name and password. InfoSphere DataStage encrypts the password when you use the auto-generate option.
    - Specify the name of the server hosting Teradata.
  - Ensure column metadata has been specified for the read.
Stage Page

The General tab allows you to specify an optional description of the stage. The Advanced tab allows you to specify how the stage executes. The NLS Map tab appears if you have NLS enabled on your system, it allows you to specify a character set map for the stage.

Advanced Tab

This tab allows you to specify the following:

- **Execution Mode.** The stage can execute in parallel mode or sequential mode. In parallel mode the data is processed by the available nodes as specified in the Configuration file, and by any node constraints specified on the Advanced tab. In Sequential mode the data is processed by the conductor node.

- **Combinability mode.** This is Auto by default, which allows IBM InfoSphere DataStage to combine the operators that underlie parallel stages so that they run in the same process if it is sensible for this type of stage.

- **Preserve partitioning.** You can select **Set** or **Clear.** If you select **Set** read operations will request that the next stage preserves the partitioning as is (the Preserve partitioning field is not visible unless the stage has an output links).

- **Node pool and resource constraints.** Select this option to constrain parallel execution to the node pool or pools, resource pool or pools (or both) specified in the grid. The grid allows you to make choices from drop down lists populated from the Configuration file.

- **Node map constraint.** Select this option to constrain parallel execution to the nodes in a defined node map. You can define a node map by typing node numbers into the text box or by clicking the browse button to open the Available Nodes dialog box and selecting nodes from there. You are effectively defining a new node pool for this stage (in addition to any node pools defined in the Configuration file).

NLS Map

The NLS Map tab allows you to define a character set map for the Teradata Enterprise stage. This overrides the default character set map set for the project or the job. You can specify that the map be supplied as a job parameter if required.

Input Page

The Input page allows you to specify details about how the Teradata Enterprise stage writes data to a Teradata database. The Teradata Enterprise stage can have only one input link writing to one table.

The General tab allows you to specify an optional description of the input link. The Properties tab allows you to specify details of exactly what the link does. The Partitioning tab allows you to specify how incoming data is partitioned before being written to the database. The Columns tab specifies the column definitions of incoming data. The Advanced tab allows you to change the default buffering settings for the input link.

Details about Teradata Enterprise stage properties, partitioning, and formatting are given in the following sections.
Input Link Properties Tab

The Properties tab allows you to specify properties for the input link. These dictate how incoming data is written and where. Some of the properties are mandatory, although many have default settings. Properties without default settings appear in the warning color (red by default) and turn black when you supply a value for them.

The following table gives a quick reference list of the properties and their attributes. A more detailed description of each property follows.

Table 10. Input link properties and values

<table>
<thead>
<tr>
<th>Category/Property</th>
<th>Values</th>
<th>Default</th>
<th>Required?</th>
<th>Dependent of</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target/Table</td>
<td>Table_Name</td>
<td>N/A</td>
<td>Y</td>
<td>N/A</td>
</tr>
<tr>
<td>Target/Primary Index</td>
<td>Columns List</td>
<td>N/A</td>
<td>N</td>
<td>Table</td>
</tr>
<tr>
<td>Target/Select List</td>
<td>List</td>
<td>N/A</td>
<td>N</td>
<td>Table</td>
</tr>
<tr>
<td>Target/Write Mode</td>
<td>Append/Create/Replace/Truncate</td>
<td>Append</td>
<td>Y</td>
<td>N/A</td>
</tr>
<tr>
<td>Connection/DB Options</td>
<td>String</td>
<td>N/A</td>
<td>Y</td>
<td>N/A</td>
</tr>
<tr>
<td>Connection/Database</td>
<td>Database Name</td>
<td>N/A</td>
<td>N</td>
<td>N/A</td>
</tr>
<tr>
<td>Connection/Server</td>
<td>Server Name</td>
<td>N/A</td>
<td>Y</td>
<td>N/A</td>
</tr>
<tr>
<td>Options/Close Command</td>
<td>Close Command</td>
<td>500</td>
<td>N</td>
<td>Insert SQL</td>
</tr>
<tr>
<td>Options/Open Command</td>
<td>Open Command</td>
<td>False</td>
<td>N</td>
<td>N/A</td>
</tr>
<tr>
<td>Options/Silently Drop Not in Tables</td>
<td>True/False</td>
<td>False</td>
<td>Y</td>
<td>N/A</td>
</tr>
<tr>
<td>Options/Default String Length</td>
<td>String Length</td>
<td>32</td>
<td>N</td>
<td>N/A</td>
</tr>
<tr>
<td>Options/Truncate Column Names</td>
<td>True/False</td>
<td>False</td>
<td>Y</td>
<td>N/A</td>
</tr>
<tr>
<td>Options/Progress Interval</td>
<td>Number</td>
<td>100000</td>
<td>N</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Target Category

The following are the Target categories.

Table

Specify the name of the table to write to. The table name must be a valid Teradata table name. Table has two dependent properties:
- **Select List.** Specifies a list that determines which columns are written. If you do not supply the list, the Teradata Enterprise stage writes to all columns. Do not include formatting characters in the list.

- **Primary Index.** Specify a comma-separated list of column names that will become the primary index for tables. Format the list according to Teradata standards and enclose it in single quotes.

  For performance reasons, the data set should not be sorted on the primary index. The primary index should not be a smallint, or a column with a small number of values, or a high proportion of null values. If no primary index is specified, the first column is used. All the considerations noted above apply to this case as well.

**Write Mode**

Select from the following:

- **Append.** Appends new records to the table. The database user must have TABLE CREATE privileges and INSERT privileges on the table being written to. This is the default.

- **Create.** Creates a new table. The database user must have TABLE CREATE privileges. If a table exists of the same name as the one you want to create, the data flow that contains Teradata terminates in error.

- **Replace.** Drops the existing table and creates a new one in its place; the database user must have TABLE CREATE and TABLE DELETE privileges. If a table exists of the same name as the one you want to create, it is overwritten. Note that you cannot create or replace a table that has primary keys, you should not specify primary keys in your metadata.

- **Truncate.** Retains the table attributes, including the table definition, but discards existing records and appends new ones. The database user must have DELETE and INSERT privileges on the table.

**Connection Category**

The following are the Connection categories.

**DB Options**

Specify a user name and password for connecting to Teradata in the form:

```
<user = <user >,
password= <password
workdbwork_database 
> [SessionsPerPlayer = <num_sessions>]
[RequestedSessions = <num_requested>]
[synctimeout = <timeout _in_secs>]
```

The value of sessionsperplayer determines the number of connections each player has to Teradata. Indirectly, it also determines the number of players. The number selected should be such that (sessionsperplayer * number of nodes * number of players per node) equals the total requested sessions. The default is 2.

If the user does not have CREATE privileges on the default database, the workdb option allows the user to specify an alternate database where the error tables and work table will be created.

Setting the value of sessionsperplayer too low on a large system can result in so many players that the step fails due to insufficient resources. In that case, sessionsperplayer should be increased.
The value of the optional requestedsessions is a number between 1 and the number of vprocs in the database. The default is the maximum number of available sessions.

synctimeout specifies the time that the player slave process waits for the control process. The default is 20 seconds.

IBM InfoSphere DataStage does not encrypt the password when you use this option.

**DB Options Mode**

If you select Auto-generate for this property, InfoSphere DataStage will create a DB Options string for you. If you select User-defined, you have to edit the DB Options property yourself. When Auto-generate is selected, there are two dependent properties:

- **User**. The user name to use in the auto-generated DB options string.
- **Password**. The password to use in the auto-generated DB options string.

InfoSphere DataStage encrypts the password.

**Database**

By default, the write operation is carried out in the default database of the Teradata user whose profile is used. If no default database is specified in that user’s Teradata profile, the user name is the default database. If you supply the database name, the database to which it refers must exist and you must have necessary privileges.

**Server**

Specify the name of a Teradata server.

**Options Category**

The following are the Options categories.

**Close Command**

Specify a Teradata command to be parsed and executed by Teradata on all processing nodes after the table has been populated.

**Open Command**

Specify a Teradata command to be parsed and executed by Teradata on all processing nodes before the table is populated.

**Silently Drop Columns Not in Table**

Specifying True causes the stage to silently drop all unmatched input columns; otherwise the job fails.

**Default String Length**

Specify the maximum length of variable-length raw or string columns. The default length is 32 bytes. The upper bound is slightly less than 32 KB.
Truncate Column Names

Specify whether the column names should be truncated to 30 characters or not.

Progress Interval

By default, the stage displays a progress message for every 100,000 records per partition it processes. Specify this option either to change the interval or to disable the message. To change the interval, specify a new number of records per partition. To disable the messages, specify 0.

Partitioning Tab

The Partitioning tab allows you to specify details about how the incoming data is partitioned or collected before it is written to the Teradata database. It also allows you to specify that the data should be sorted before being written.

By default the stage partitions in Auto mode. This attempts to work out the best partitioning method depending on execution modes of current and preceding stages and how many nodes are specified in the Configuration file.

If the Teradata Enterprise stage is operating in sequential mode, it will first collect the data before writing it to the file by using the default Auto collection method.

The Partitioning tab allows you to override this default behavior. The exact operation of this tab depends on:

- Whether the Teradata Enterprise stage is set to execute in parallel or sequential mode.
- Whether the preceding stage in the job is set to execute in parallel or sequential mode.

If the Teradata Enterprise stage is set to execute in parallel, then you can set a partitioning method by selecting from the Partition type drop-down list. This will override any current partitioning.

If the Teradata Enterprise stage is set to execute in sequential mode, but the preceding stage is executing in parallel, then you can set a collection method from the Collector type drop-down list. This will override the default collection method.

The following partitioning methods are available:

- **(Auto)**. IBM InfoSphere DataStage attempts to work out the best partitioning method depending on execution modes of current and preceding stages and how many nodes are specified in the Configuration file. This is the default partitioning method for the Teradata Enterprise stage.
- **Entire**. Each file written to receives the entire data set.
- **Hash**. The records are hashed into partitions based on the value of a key column or columns selected from the Available list.
- **Modulus**. The records are partitioned by using a modulus function on the key column selected from the Available list. This is commonly used to partition on tag columns.
- **Random**. The records are partitioned randomly, based on the output of a random number generator.
- **Round Robin**. The records are partitioned on a round robin basis as they enter the stage.
• **Same.** Preserves the partitioning already in place. This is the default for Teradata Enterprise stages.

• **Range.** Divides a data set into approximately equal size partitions based on one or more partitioning keys. Range partitioning is often a preprocessing step to performing a total sort on a data set. Requires extra properties to be set. Access these properties by clicking the properties button.

The following Collection methods are available:

• **(Auto).** This is the default collection method for Teradata Enterprise stages. Normally, when you are using Auto mode, InfoSphere DataStage will eagerly read any row from any input partition as it becomes available.

• **Ordered.** Reads all records from the first partition, then all records from the second partition, and so on.

• **Round Robin.** Reads a record from the first input partition, then from the second partition, and so on. After reaching the last partition, the operator starts over.

• **Sort Merge.** Reads records in an order based on one or more columns of the record. This requires you to select a collecting key column from the **Available** list.

The **Partitioning** tab also allows you to specify that data arriving on the input link should be sorted before being written to the database. The sort is always carried out within data partitions. If the stage is partitioning incoming data the sort occurs after the partitioning. If the stage is collecting data, the sort occurs before the collection. The availability of sorting depends on the partitioning or collecting method chosen (it is not available with the default Auto methods).

Select the check boxes as follows:

• **Perform Sort.** Select this to specify that data coming in on the link should be sorted. Select the column or columns to sort on from the **Available** list.

• **Stable.** Select this if you want to preserve previously sorted data sets. This is the default.

• **Unique.** Select this to specify that, if multiple records have identical sorting key values, only one record is retained. If stable sort is also set, the first record is retained.

If NLS is enabled an additional button opens a dialog box allowing you to select a locale specifying the collate convention for the sort.

You can also specify sort direction, case sensitivity, whether sorted as ASCII or EBCDIC, and whether null columns will appear first or last for each column. Where you are using a keyed partitioning method, you can also specify whether the column is used as a key for sorting, for partitioning, or for both. Select the column in the **Selected** list and right-click to invoke the shortcut menu.

### Output Page

The **Output page** allows you to specify details about how the Teradata Enterprise stage reads data from a Teradata database. The Teradata Enterprise stage can have only one output link.

The **General** tab allows you to specify an optional description of the output link. The **Properties** tab allows you to specify details of exactly what the link does. The
Columns tab specifies the column definitions of the data. The Advanced tab allows you to change the default buffering settings for the output link.

Details about Teradata Enterprise stage properties are given in the following sections.

Output Link Properties Tab

The Properties tab allows you to specify properties for the output link. These dictate how incoming data is read and from what table. Some of the properties are mandatory, although many have default settings. Properties without default settings appear in the warning color (red by default) and turn black when you supply a value for them.

The following table gives a quick reference list of the properties and their attributes. A more detailed description of each property follows.

Table 11. Output link properties and values

<table>
<thead>
<tr>
<th>Category/Property</th>
<th>Values</th>
<th>Default</th>
<th>Mandatory?</th>
<th>Dependent of</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source/Read Method</td>
<td>Table/Auto-generated SQL/User-defined SQL</td>
<td>Table</td>
<td>Y</td>
<td>N/A</td>
</tr>
<tr>
<td>Source/Table</td>
<td>Table Name</td>
<td>Y (if Read Method = Table or Auto-generated SQL)</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>Source/Select List</td>
<td>List</td>
<td>N/A</td>
<td>N</td>
<td>Table</td>
</tr>
<tr>
<td>Source/Where Clause</td>
<td>Filter</td>
<td>N/A</td>
<td>N</td>
<td>Table</td>
</tr>
<tr>
<td>Source/Query</td>
<td>SQL query</td>
<td>N/A</td>
<td>Y (if Read Method = User-defined SQL or Auto-generated SQL)</td>
<td>N/A</td>
</tr>
<tr>
<td>Connection/DB Options</td>
<td>String</td>
<td>N/A</td>
<td>Y</td>
<td>N/A</td>
</tr>
<tr>
<td>Connection/Database</td>
<td>Database Name</td>
<td>N/A</td>
<td>N</td>
<td>N/A</td>
</tr>
<tr>
<td>Connection/Server</td>
<td>Server Name</td>
<td>N/A</td>
<td>Y</td>
<td>N/A</td>
</tr>
<tr>
<td>Options/Close Command</td>
<td>String</td>
<td>N/A</td>
<td>N</td>
<td>N/A</td>
</tr>
<tr>
<td>Options/Open Command</td>
<td>String</td>
<td>N/A</td>
<td>N</td>
<td>N/A</td>
</tr>
<tr>
<td>Options/Progress Interval</td>
<td>Number</td>
<td>100000</td>
<td>N</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Source Category

The following are the Source categories.
**Read Method**

Select Table to use the Table property to specify the read (this is the default). Select Auto-generated SQL this to have IBM InfoSphere DataStage automatically generate an SQL query based on the columns you have defined and the table you specify in the Table property. You must select the Query property and select Generate from the right-arrow menu to actually generate the statement. Select User-defined SQL to define your own query.

**Table**

Specifies the name of the Teradata table to read from. The table must exist, and the user must have the necessary privileges to read it.

The Teradata Enterprise stage reads the entire table, unless you limit its scope by means of the Select List, Where suboptions, or both:

- **Select List.** Specifies a list of columns to read. The items of the list must appear in the same order as the columns of the table.
- **Where Clause.** Specifies selection criteria to be used as part of an SQL statement's WHERE clause. Do not include formatting characters in the query.

These dependent properties are only available when you have specified a Read Method of Table rather than Auto-generated SQL.

**Query**

This property is used to contain the SQL query when you choose a Read Method of User-defined query or Auto-generated SQL. If you are using Auto-generated SQL you must select a table and specify some column definitions, then select Generate from the right-arrow menu to have InfoSphere DataStage generate the query.

**Connection Category**

The following are the Connection categories.

**DB Options**

Specify a user name and password for connecting to Teradata in the form:

```
<user> = <user>,
password= <password>
workdwork_database>
[SessionsPerPlayer = <num_sessions>]
[RequestedSessions = <num_requested>]
[synctimeout = <timeout _in_secs>]
```

If the user does not have CREATE privileges on the default database, the workdb option allows the user to specify an alternate database where the error tables and work table will be created.

The value of sessionsperplayer determines the number of connections each player has to Teradata. Indirectly, it also determines the number of players. The number selected should be such that (sessionsperplayer * number of nodes * number of players per node) equals the total requested sessions. The default is 2.
Setting the value of sessionsperplayer too low on a large system can result in so many players that the step fails due to insufficient resources. In that case, sessionsperplayer should be increased.

The value of the optional requestedsessions is a number between 1 and the number of vprocs in the database. The default is the maximum number of available sessions.

synctimeout specifies the time that the player slave process waits for the control process. The default is 20 seconds.

IBM InfoSphere DataStage does not encrypt the password when you use this option.

**DB Options Mode**

If you select Auto-generate for this property, InfoSphere DataStage will create a DB Options string for you. If you select User-defined, you have to edit the DB Options property yourself. When Auto-generate is selected, there are two dependent properties:

- **User.** The user name to use in the auto-generated DB options string.
- **Password.** The password to use in the auto-generated DB options string. InfoSphere DataStage encrypts the password.

**Database**

By default, the read operation is carried out in the default database of the Teradata user whose profile is used. If no default database is specified in that user's Teradata profile, the user name is the default database. This option overrides the default.

If you supply the database name, the database to which it refers must exist and you must have the necessary privileges.

**Server**

Specify the name of a Teradata server.

**Options Category**

The following are the Options categories.

**Close Command**

Optionally specifies a Teradata command to be run once by Teradata on the conductor node after the query has completed.

**Open Command**

Optionally specifies a Teradata command run once by Teradata on the conductor node before the query is initiated.

**Progress Interval**

By default, the stage displays a progress message for every 100,000 records per partition it processes. Specify this option either to change the interval or to disable
the message. To change the interval, specify a new number of records per partition. To disable the messages, specify 0.
Chapter 6. Teradata Load stage

The Teradata Load stage is a passive stage that loads streams of tabular data into tables of a target Teradata database.

When you use IBM InfoSphere DataStage to access Teradata databases, you can choose from a collection of connectivity options. For most new jobs, use the Teradata Connector stage, which offers better functionality and performance than the Teradata Load stage.

If you have jobs that use the Teradata Load stage and want to use the connector, use the Connector Migration Tool to migrate jobs to use the connector.

This chapter describes the following topics:

- "Functionality of Teradata Load Stages"
- "Using Teradata Load"
- "Teradata Input Link"
- "Teradata Output Link"
- "Data Type Support"

Introduction to Teradata Load

Teradata Load generates, and optionally automatically executes, the Teradata commands to load an empty Teradata table in bulk with data from input links. It supports FastLoad, the Teradata Load command-line utility. This utility is faster than MultiLoad for this type of operation. This stage can have any number of input and output links. Reference links have no meaning in the context of the Teradata Load stage and are not allowed.

Input Links

Input links specify the data you are writing, which is a stream of rows to be loaded into a Teradata table. The Teradata Load stage generates a control and data file, which use the Teradata FastLoad utility to load into a single table. The data file contains formatted rows and columns to be loaded into a single table on a Teradata target database.

Output Links

Output links specify the data you are reading, which is a formatted file exported from a Teradata database by using the Basic Teradata Query (BTEQ) utility.

Installing Teradata Load

Before installing the stage, consult Teradata documentation for any specific configuration requirements.

Install the Teradata client software on IBM InfoSphere DataStage server machine. Use the Teradata client software only for automatic execution of load commands. For manual execution of FastLoad commands in batch mode, no Teradata software is necessary to create control and data files. To execute FastLoad later, Teradata software is required.
Ensure that the Teradata server is running for the automatic execution of FastLoad load commands from the stage or for manual execution of FastLoad commands at the DOS prompt.

### Configuring the Environment for Teradata Stages

**About this task**

Some Teradata stages require the setting of environment variables in order to work correctly on a UNIX platform. To add or change an environment variable, include any environment variables in the `dsenv` file. The following table identifies the specific environment variables that are required.

<table>
<thead>
<tr>
<th>For...</th>
<th>Set the following environment variable on the IBM InfoSphere DataStage server machine...</th>
</tr>
</thead>
</table>
| Teradata API stage             | LD_LIBRARY_PATH
|                                | PATH
| Teradata Load stage            | LD_LIBRARY_PATH
|                                | PATH
| Teradata MultiLoad/TPump/FastExport stage | LD_LIBRARY_PATH
|                                | PATH

The name of one particular environment variable, referred to as `LD_LIBRARY_PATH` above, differs depending on the platform. See the following to determine the correct name to use for your environment.

- If the platform is IBM AIX, use `LIBPATH`.
- If the platform is HP_UX, use `SHLIB_PATH`.
- If the platform is Linux, Solaris, or Tru64, use `LD_LIBRARY_PATH`.

### Teradata Stages and the Parallel Canvas

Some connectivity stages were designed for server jobs but can run on the parallel canvas. The default for **all** stages is Sequential. "In Parallel" mean you can set it to run in parallel, but this is not the default. Use the following table to determine which connectivity server stages are available on the parallel canvas.

<table>
<thead>
<tr>
<th>Connectivity Stage</th>
<th>Available on the Parallel Canvas in Windows</th>
<th>Available on the Parallel Canvas in UNIX</th>
<th>Used as a Source or a Target or for Processing</th>
<th>Runs Sequentially or In Parallel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teradata API</td>
<td>Yes</td>
<td>Yes</td>
<td>Source or Target</td>
<td>Source: Sequential; Target: In Parallel</td>
</tr>
<tr>
<td>Teradata Load</td>
<td>Yes</td>
<td>Yes</td>
<td>Target</td>
<td>Sequential</td>
</tr>
</tbody>
</table>
Table 13. Availability on the Parallel Canvas (continued)

<table>
<thead>
<tr>
<th>Connectivity Stage</th>
<th>Available on the Parallel Canvas in Windows</th>
<th>Available on the Parallel Canvas in UNIX</th>
<th>Used as a Source or a Target or for Processing</th>
<th>Runs Sequentially or In Parallel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teradata MultiLoad/TPump/FastExport</td>
<td>Yes</td>
<td>Yes</td>
<td>Source or Target</td>
<td>Sequential</td>
</tr>
<tr>
<td>Teradata connector</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Sequential and Parallel</td>
</tr>
</tbody>
</table>

Mapping String Data

The purpose of the NONE map on the server canvas is to turn off mapping of string data in any stage in which the map is set, i.e., to pass the data through verbatim. This feature is handled differently on the parallel canvas. When you define string data (char, varchar, etc.), there is an additional field in the stage’s Columns grid called Extended. This can be set to blank or Unicode. If this option is set to blank, no mapping occurs (i.e. “NONE”); the map specified on the NLS tab is ignored. If this option is set to Unicode, the NLS map is applied. In order to read or write Japanese data, for example, set Extended to Unicode. When the job compiler detects this combination (char, varchar, etc. and Unicode), it generates the appropriate runtime code.

Functionality of Teradata Load Stages

The Teradata Load stage has the following functionality:

- Support for data files which exceed the 2 GB file size limit for 64-bit file systems.
- Support for NLS (National Language Support).
- Generation, and optional automatic execution, of the Teradata commands to load the database with data from input links.
- Support for loading from a stream input link to provide rows of data into the target table.
- Support for FastLoad, the Teradata Load utility for loading data.
- Support for a control file that contains Teradata commands to load or reload a database.
- Two load modes: manual or automatic.
- Load parameters to control the load process.
- Automatically drops and creates specified target tables.

The following functionality is not supported:

- Compatibility with IBM InfoSphere DataStage releases before 7.0
- Reference output links
- Metadata importing
- Support for stored procedures
- Native data browsing
- Supports reject row handling
Using Teradata Load

About this task

Double click the Teradata Load icon, right click the Teradata Load icon and select Properties, or choose Properties from the Edit menu. The Terabulk stage dialog box opens. This dialog box has up to three pages:

- The Stage page displays the name of the stage you are editing, the General tab and optionally the NLS tab.
- The Input page specifies the information necessary to load data in a Teradata database using FastLoad. This page also specifies the associated column definitions. See “Teradata Input Link.”
- The Output page specifies the information necessary to import data from a Teradata database into InfoSphere DataStage. This page also specifies the associated column definitions. See “Teradata Output Link” on page 107.

About the Stage Page

The Stage page consists of the Stage name, the General tab, and NLS tab (optional). The General tab on the Stage page opens by default.

- Stage name. The name of the stage you are editing.

The General Tab

The following field is in the General tab:

- Description. Optionally, a description of the purpose of the Teradata Load stage.

Defining Character Set Mapping

About this task

You can define a character set map for a stage. Do this from the NLS tab that appears on the Stage page. The NLS page appears only if you have installed NLS.

Enter information for the following button and fields, if appropriate:

- Map name to use with stage. The default character set map is defined for the project or the job. You can change the map by selecting a map name from the list.
- Use Job Parameter... . The parameter values for the job. Use the format #Param#, where Param is the name of the job parameter. The string #Param# is replaced by the job parameter when the job is run.
- Show all maps. A list of all the maps that are shipped with InfoSphere DataStage.
- Loaded maps only. A list of only the maps that are currently loaded.

Teradata Input Link

When you write data to a Teradata database, Teradata Load has an input link.

Loading Data

The Teradata Load stage uses the FastLoad utility to load data into a Teradata table. FastLoad is a command-driven utility that uses multiple sessions to quickly load large amounts of data into an empty table on a Teradata database.
If Teradata Load has an input link, it generates data to load into a Teradata database table by creating a control and data file for FastLoad. An IBM InfoSphere DataStage job generates one control file for each input link to each instance of Teradata Load.

The column names and data types of the data are those of the InfoSphere DataStage metadata and must match those of the Teradata table to be loaded or cleared. You can optionally drop and recreate the Teradata table.

The source data for a FastLoad operation comes from the data file generated by the stage. The target table on the Teradata database must be empty, with no defined secondary indexes. Each FastLoad job loads one table into the Teradata database. When using FastLoad to load more than one table, you must submit multiple FastLoad jobs, one for each table.

FastLoad accepts both FastLoad commands and a subset of Teradata SQL statements. The FastLoad commands perform two types of activities:
- Session control commands begin and end FastLoad sessions.
- Data handling commands establish and define a FastLoad operation.

### Loading Modes

You can run FastLoad manually in batch mode or automatically from the stage. Set **Load Method** to **Manual** or **Invoke FastLoad** to do this. For additional information, see “The Load Tab” on page 103.

#### Manual Loading

You may want to run FastLoad manually in batch mode for the following reasons:
- If the FastLoad client software resides on IBM InfoSphere DataStage server, you can use the **Load Method** to run FastLoad in batch mode after the data and control files have been written. If, however, FastLoad resides on another client or server, you must copy the control and data files to that machine and run FastLoad from there. You can use before- and after-job routines to do this.
- If you need to modify a parameter in the control file before running FastLoad.
- You must also consider the length of time and the system resources FastLoad consumes when you run it by using the stage.

#### Running FastLoad Manually in Batch Mode

**About this task**

To run FastLoad manually in batch mode:

**Procedure**

1. Copy the control and data files to the machine where FastLoad resides.
2. Change the directory to the one specified by **Output Files Path**. (If no value is specified in **Output Files Path**, change to IBM InfoSphere DataStage project directory.)
3. Execute the following command:

   ```
   fastload < controlfile.fl
   controlfile is the value you specify in Control File.
   ```
Automatic Loading

If you want to run FastLoad from the stage, use the output file from the FastLoad session in the directory path of the control and data files to verify that the load has succeeded. It has a similar name as the control file with the \_floutput\_txt extension appended, for example:

controlfile\_floutput.txt

**Note:** Automatic execution of load commands works only when the Teradata client resides on the same machine as IBM InfoSphere DataStage job.

Troubleshooting

Examine the Teradata tables to see why the FastLoad failed. FastLoad stores the input data records related to constraint violations, conversion errors, unavailable Access Module Process (AMP) conditions, and unique primary index violations in the two error tables that you specify in Error Table 1 and Error Table 2. See ["The Load Tab" on page 103](#).

This output contains the end-of-job status report. FastLoad discards all records that produce a duplicate row error, but includes the total number of duplicate rows encountered. It also includes the total records in each error table in the end-of-job status report.

Input Data

Teradata Load generates data files as formatted data, which is input data that conforms to the format of data from a Teradata database source, such as a BTEQ EXPORT file. Each record has a two-byte data length field, a variable-length indicator byte field, a variable-length input data field, and a one-byte end-of-record delimiter field.

SET RECORD Command

The stage uses the SET RECORD command to specify the format of the input data as FORMATTED. Indicators, which let Teradata Load deal with null data (that is, data with no values) are always generated. Thus, null values can be easily loaded into Teradata, and NULLIF qualifiers are not required. This is forced by always adding the INDICATORS keyword to the BEGIN LOADING command. The number of fields in each record determines how many bytes contain null indicators. FastLoad expects the first bytes of the record to contain an indicator bit for each record field. Consult [Teradata FastLoad Reference](#) for more information on indicator bits.

DEFINE Command

The DEFINE command describes the fields in a record of input data that are inserted in the FastLoad table. Also, it identifies the name of the input data source. Teradata Load uses the DEFINE command to identify the input data source.

INSERT Statement

To describe the fields in a record of input data the stage uses the INSERT table.* syntax. This retrieves a list of field names from the referenced table. When you use this format of the INSERT statement, FastLoad constructs a list of field names from the table definition. During the insert operation, FastLoad gets the field names and their data types from the CREATE TABLE statement used to define the table and from the table definition. The field name definitions are established in the order in which columns are defined in the CREATE TABLE statement. Therefore, the fields in each data record must be in the same order as the columns in the definition of the table.
Note: The Teradata INSERT statement for Teradata Load does not support all formats of Unicode data.

Privileges
The user who is logged in must have the required privileges and rights to run the FastLoad job. Consult the FastLoad documentation for a list of these privileges.

About the Input Page
The Input page has an Input name field, a Columns... button, and General, MultiLoad/TPump, TPUMP, Limit Settings, Script, and Columns tabs.

- **Input name.** The name of the input link. Choose the link you want to edit from the Input name list. This list displays all the input links to the Teradata Load stage.
- **Columns...** A brief list of the columns designated on the input link. As you enter detailed metadata on the Columns tab, you can leave this list displayed.

The General Tab
Use the General tab on the Input page to identify the target of the import into a Teradata table.

The General tab contains the following fields:
- **Server.** The name of a Teradata Director Program (TDP). Optional.
- **User ID.** A name that identifies the user. The user must have the necessary privileges to write to the database. Required.
- **Password.** The password associated with the user name. Required.
- **Account.** The account associated with the user name. Optional.
- **Database.** The name of the database to be loaded or updated. Optional.
- **Client character set.** The Teradata client character set to use when connecting to the Teradata server. The default value is Default.
- **Description.** A description of the link. Optional.

The Load Tab
Use the Load tab to provide information about the table and the load method.

The Load tab contains the following fields:
- **Table.** The name of the FastLoad target table to receive the data from the client system. This name is used in FastLoad statements, such as the INSERT tablename.* statement. This statement generates a list of field names from the definition of the table. Required.
- **Error Table 1.** The name of the FastLoad error Table 1 for rows ignored by Teradata because of an error. This table contains records that were rejected because of an error other than unique primary index or duplicate row violation. The stage ensures this is a new table by doing a DROP TABLE first. This name is used by the BEGIN LOADING statement. Required.
- **Error Table 2.** The name of the FastLoad error Table 2 for rows ignored by Teradata because of an error. This table contains records that violate the unique primary index constraint. The stage ensures this is a new table by doing a DROP TABLE first. This name is used in the BEGIN LOADING statement. Required. FastLoad automatically drops error tables that contain no rows when END LOADING finishes executing.
- **Pre-load Table Action.** The action performed on the table prior to loading it. The stage generates the necessary DROP and CREATE TABLE statements to
reestablish the table. The IBM InfoSphere DataStage metadata is used in each of these statements that are created. The options are:

- **Clear the table.** Does not drop the table but deletes all rows from the table.
- **Drop and create the table.** Drops the table and recreates it before loading.
- **User-defined DDL.** Enables the User-defined DDL tab on the Script tab. The User-defined DDL tab allows you to customize the DROP TABLE and CREATE TABLE statements.

**Drop Error Tables Before Load.** The disposition of the error tables. If Drop Error Tables Before Load is selected (which is the default), the generated script drops the error tables before loading begins. If Drop Error Tables Before Load is not selected, the stage assumes one of the following:

- There are no error tables to drop. Therefore, the stage does not put statements to drop the error tables in the generated script. If error tables preexist, FastLoad fails.
- You are attempting a restart. Therefore the error tables must preexist from a previous run.

**Load Method.** The method for loading the data. The options are:

- **Manual.** The files to perform a FastLoad are generated, but no loading is performed by the stage.
- **Invoke FastLoad.** FastLoad is run after creating the control and data files. Output from the FastLoad sessions is saved in a file.

**Time Data Type.** The data type of time columns expected by the stage in the Teradata database. The options are:

- **REAL.** The stage expects time columns to be defined as REAL in the Teradata database. The default value.

  **Note:** Time is encoded for Teradata as \((\text{hour} \times 10000 + \text{minute} \times 100 + \text{second})\) where \(\text{second}\) might include a fractional value.

- **TIME(n).** The stage expects time columns to be defined as TIME(n), where \(n\) is the Scale value in the range of 0 - 6. 0 - 6 represents the fractional seconds precision. The Length value for the time column must be 8 if Scale equals 0 or the sum of 9 and Scale if Scale is greater than 0.

**Timestamp Data Type.** The data type of timestamp columns expected by the stage in the Teradata database. The options are:

- **CHAR(19).** The stage expects timestamp columns to be defined as CHAR(19) in the Teradata database. The default value.
- **TIMESTAMP(n).** The stage expects timestamp columns to be defined as TIMESTAMP(n), where \(n\) is the Scale value in the range of 0 - 6, which represents the fractional seconds precision. The Length value for the timestamp column must be 19 if Scale equals 0 or the sum of 20 and Scale if Scale is greater than 0.

**The Files Tab**

Use the Files tab to provide information about the support files.

The Files tab contains the following fields:

- **Report File.** The name of the report file. If omitted, the default is controlfile_floutput.txt.
- **Control File.** A local name for the FastLoad control file. If the control file does not exist, it is created. If the control file already exists, it is truncated. If omitted, the default is DATABASE_TABLE.f1. If Database is also omitted, the default is TABLE.f1.
- **Data File.** The name of the FastLoad data file, including the .dat extension. If omitted, the default is DATABASE_TABLE.dat. If Database is also omitted, the default is TABLE.dat. The value of this property is used in the DEFINE statement. Table definitions are used to define data so that the stage uses the DEFINE statement only with the FILE keyword.

- **Output Files Path.** The directory path for FastLoad control and data files. If omitted, the stage stores the files in the IBM InfoSphere DataStage project directory on the InfoSphere DataStage server.

- ... (ellipsis). A button that opens a dialog box in which you can browse for the directory path for FastLoad control and data files.

- **Use Named Pipe.** A control that determines whether a named pipe is used for autoload. This property is available only if Load Method is *Invoke FastLoad.*
  - If you want rows to be streamed to FastLoad via a named pipe, select Use Named Pipe. If Load Method is Manual, rows are written to a data file. The name and location of the pipe are determined by Data File and Output Files Path.
  - If you do not want to use a named pipe, clear Use Named Pipe. Rows are written to a data file. This is the default.

- **Overwrite Existing Data File.** The procedure for handling an existing data file. This property is available only if Use Named Pipe is cleared.
  - If you want to replace the current data file each time the job is run, select Overwrite Existing Data File. This is the default.
  - If you want to ignore the input rows and start the load utility, which must load from an existing data file, clear Overwrite Existing Data File.

- **Create Data File with Default Permissions (Unix only).** The permissions assigned to the data file when it is created by the stage. This property applies only to UNIX.
  - If you want to use the Default file mode creation mask when creating the data file on a UNIX system, select Create Data File with Default Permissions (Unix only). This is the default.
  - If you want to create the data file with the standard set of permissions that all loaders use, clear Create Data File with Default Permissions (Unix only).

- **Delete Data File After Load.** The disposition of a data file after a load. This property is available only if Use Named Pipe is cleared.
  - If you want to delete the data file after the data is loaded successfully, select Delete Data File After Load. This is the default.
  - If you want to leave the data file intact after loading completes, clear Delete Data File After Load. This is the default.

- **Data Format.** The format in which rows are written to the data file. The options are:
  - FASTLOAD. Rows are written in the Teradata RDBMS standard format. This is the default.
  - VARTEXT. Rows are written in variable-length text record format with each field separated by a delimiter character.

- **Delimiter.** The delimiter character that separates fields in variable-length text records. This property is available only if Data Format is VARTEXT. The default delimiter is the pipe character (|).

**The Limit Settings Tab**

Use the Limit Settings tab to provide parameters to the FastLoad utility. The fields correspond directly to options in the FastLoad utility.
The **Limit Settings** tab contains the following fields:

- **Sess Max.** The number of FastLoad sessions. This property generates a SESSIONS statement, which specifies the number of FastLoad sessions to be logged on when you enter a LOGON statement. The default is 4.
- **Sess Min.** The minimum number of sessions required for the job to continue. The default of 0 uses the default value of the FastLoad utility.
- **Buffer Size.** The size of the output buffer, in kilobytes, that is used for FastLoad messages to the Teradata RDBMS. A larger buffer size reduces processing overhead by including more data in each message.
  
  The default buffer size of 0 uses the default value set for the FastLoad utility. If you specify a value greater than the maximum, the FastLoad utility responds with a warning, resets the buffer size to the default value, and continues with the FastLoad job.
- **Error Limit.** The maximum number of rows that can be rejected during a FastLoad run. This generates an ERRLIMIT statement, which limits the number of records that can be rejected while inserting data into the FastLoad table.
  
  The default of 0 does not generate an ERRLIMIT statement and uses the Teradata default.
  
  If you expect no errors in the input data, set the error limit value to 1. The job terminates when any record causes an error.
- **Checkpoint.** The number of rows transmitted to the Teradata RDBMS between checkpoints. This generates a CHECKPOINT option on the BEGIN LOADING command. The default of 0 does not generate a CHECKPOINT option.
- **Start Row.** The row number for FastLoad to start loading from the data file. The default of 0 means processing begins with the first record.

  **Note:** This property generates a RECORD statement, which defines the records of the input data source at which FastLoad processing starts and ends.
- **End Row.** The row number for FastLoad to stop loading from the data file. The default of 0 means processing ends with the last record.

  The value for this property must be greater than or equal to that of the start record.

  **Note:** This property generates a RECORD statement, which defines the records of the input data source at which FastLoad processing starts and ends.
- **Sleep.** The number of minutes that the FastLoad utility should wait before retrying a logon operation when the maximum number of concurrent load operations are in progress. The default 0 does not generate a SLEEP statement and uses the Teradata default.
- **Tenacity.** The maximum number of hours that the FastLoad utility should continue trying to log on when the maximum number of concurrent load operations are in progress. The default 0 does not generate a TENACITY statement and uses the Teradata default.

See your FastLoad documentation for descriptions of these settings.

**The Script Tab**

Use the **Script** tab to view the generated FastLoad script.

The **Script** tab contains the **Generated**, **Generated DDL**, and **User-defined DDL** tabs.

- **Generated.** The generated FastLoad script.
• **Generate INSERT Column Names.** The disposition of the target column names.
  - If you want the generated INSERT statement to contain the target column names, select **Generate INSERT Column Names.** Select **Generate INSERT Column Names** if the order of the columns on the Columns tab does not match the order of the columns in the table or if the data types do not match precisely. The names of the columns on the Columns tab must match the names of the columns in the table.
  - If you do not want the generated INSERT statement to contain the target column names, do not select **Generate INSERT Column Names.**

• **Generated DDL.** The generated DROP TABLE and CREATE TABLE statements. **DROP Statement** displays the generated DROP TABLE statement for dropping the target table. **CREATE Statement** displays the generated CREATE TABLE statement that is generated from the column metadata definitions and the information provided in the Load tab. The **Generated DDL** tab is disabled if **Pre-load Table Action** is set to **Clear the Table.**

• **User-defined DDL.** The DROP TABLE and CREATE TABLE statements customized by the user. **DROP Statement** displays the user-defined DROP TABLE statement for dropping the target table. **CREATE Statement** displays the user-defined CREATE TABLE statement for creating the target table. Use the **User-defined DDL** tab to customize the CREATE TABLE statement. The **User-defined DDL** tab is disabled if **Pre-load Table Action** is set to **Clear the Table** or **Drop and create the table.**

**The Columns Tab**

Use the **Columns** tab to define the metadata for the data to be imported to the Teradata database.

You can manually enter the metadata in the grid or load it from IBM InfoSphere DataStage repository. The columns listed determine what is imported to the Teradata database.

Click **Save** to save the metadata to InfoSphere DataStage repository. Click **Load** to load the metadata from the repository.

**Teradata Output Link**

When you read data into IBM InfoSphere DataStage, Teradata Load has an output link. The output link provides a sequence of rows to read from an export file, which is created by the BTEQ export utility. The metadata for each output column in the Teradata table determines how the data is read.

BTEQ creates this intermediary export file in binary format. Use the output link to read it into InfoSphere DataStage.

Use the SET RECORDMODE ON command before exporting. The column names and data types of InfoSphere DataStage metadata must match the Teradata table.

InfoSphere DataStage can recognize files exported with BTEQ with or without indicators. The values in the columns are set to null if they are used and set.

**File Contains Indicator Bytes** controls whether the file was exported using indicator mode and whether indicator bytes are present in the file. If **File Contains Indicator Bytes** is selected and an indicator bit is set to 1, InfoSphere DataStage sets the value of the corresponding field to null when the record is loaded.
Selecting **File Contains Indicator Bytes** means the file was exported with indicators. If selected and indicator bits are not entered at the beginning of the record, Teradata Load assumes that the first field contains indicator bytes and reads the record incorrectly.

If the indicator bit is set to 0, InfoSphere DataStage loads the data specified for that field.

**About the Output Page**

The Output page has an **Output name** field, a **Columns...** button, and General and Columns tabs.

- **Output name.** The name of the output link. Choose the link you want to edit from the **Output name** list. This list displays all the output links from the Teradata Load stage.
- **Columns...**. A brief list of the columns designated on the output link. As you enter detailed metadata on the **Columns** tab, you can leave this list displayed.

**The General Tab**

Use the **General** tab on the Output page to identify the source of the import into InfoSphere DataStage.

The **General** tab contains the following fields:

- **Data File.** The name of the file to be imported, including the .dat extension. Required.
- **File Path.** The directory that contains the file to be imported. Required.
- **File Contains Indicator Bytes.** A control that determines whether the file is exported with indicators and whether indicator bytes are present in the file.

<table>
<thead>
<tr>
<th>If File Contains Indicator Bytes is</th>
<th>And an indicator bit is</th>
<th>Then InfoSphere DataStage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selected</td>
<td>Set to 1</td>
<td>Sets the value in the corresponding field to null when the record is loaded, that is, no value</td>
</tr>
<tr>
<td>Not selected</td>
<td></td>
<td>Imports the file without indicators</td>
</tr>
<tr>
<td>Selected</td>
<td>Set to 0</td>
<td>Loads the data specified for that field</td>
</tr>
</tbody>
</table>

**Note:** If **File Contains Indicator Bytes** is selected and indicator bits are not entered at the beginning of the record, the stage assumes that the first field contains indicator bytes and reads the record incorrectly.

- **Delete Data File After Import.** The disposition of a data file after an import into the InfoSphere DataStage.
  - If you want to delete the data file after the data has been read successfully, select **Delete Data File After Import**.
  - If you want to leave the data file intact after all rows have been read, clear **Delete Data File After Import**. This is the default.
- **Time Data Type.** The data type of time columns expected by the stage in the Teradata database. The options are:
- **REAL.** The stage expects time columns to be defined as REAL in the Teradata database. The default value.

  **Note:** Time is encoded for Teradata as \((\text{hour} \times 10000 + \text{minute} \times 100 + \text{second})\) where \text{second} might include a fractional value.

- **TIME(n).** The stage expects time columns to be defined as TIME(n), where \(n\) is the Scale value in the range of 0 - 6. 0 - 6 represents the fractional seconds precision. The Length value for the time column must be 8 if Scale equals 0 or the sum of 9 and Scale if Scale is greater than 0.

- **Timestamp Data Type.** The data type of timestamp columns expected by the stage in the Teradata database. The options are:
  - **CHAR(19).** The stage expects timestamp columns to be defined as CHAR(19) in the Teradata database. The default value.
  - **TIMESTAMP(n).** The stage expects timestamp columns to be defined as TIMESTAMP(n), where \(n\) is the Scale value in the range of 0 - 6, which represents the fractional seconds precision. The Length value for the timestamp column must be 19 if Scale equals 0 or the sum of 20 and Scale if Scale is greater than 0.

- **Description.** A description of the link. Optional.

**The Columns Tab**

Use the **Columns** tab to define the metadata for the data to be imported from the Teradata database.

You can manually enter the metadata in the grid or load it from the InfoSphere DataStage repository. The columns listed determine what is imported to InfoSphere DataStage.

Click **Save** to save the metadata to the InfoSphere DataStage repository. Click **Load** to load the metadata from the repository.

**Data Type Support**

Teradata Load supports mapping to and from IBM InfoSphere DataStage SQL data types and Teradata SQL data types.

All InfoSphere DataStage data types are supported except the following:
- BIGINT
- LONGVARBINARY
- LONGVARCHAR

All Teradata data types are supported except the following:
- Graphic
- Vargraphic
- Long Vargraphic

The Teradata Load stage assumes a Char or VarChar column is Unicode if the value specified in **Display** on the Columns tab is double the value specified in **Length**. Therefore, if you select the option to drop and recreate the table, the **Generated DDL** tab shows a CHARACTER SET UNICODE clause on those columns.
The following section describes DATE considerations for mapping to and from InfoSphere DataStage SQL data types and Teradata SQL data types.

**DATE Data Type Considerations**

If **DataFormat** is **FASTLOAD**, DATE data is converted from its IBM InfoSphere DataStage internal format to a Teradata internal format and stored as a Teradata DATE data type. This is the default Teradata DATE data type specification for FastLoad jobs if you do not enter a DATEFORM command.

Teradata stores each date value as an integer in a DATE data type by using the following formula:

\[(\text{year} - 1900) \times 1000 + (\text{month} \times 100) + \text{day}\]

If **DataFormat** is **V ARTEXT**, DATE is converted from its InfoSphere DataStage internal format to a character date in YYYY-MM-DD format.
Chapter 7. Teradata Multiload stage

The Teradata Multiload stage uses the Teradata MultiLoad utility or the Teradata TPump utility to load data into a Teradata database. It uses the Teradata FastExport utility to read data from a Teradata database.

When you use IBM InfoSphere DataStage to access Teradata databases, you can choose from a collection of connectivity options. For most new jobs, use the Teradata Connector stage, which offers better functionality and performance than the Teradata Multiload stage.

If you have jobs that use the Teradata Multiload stage and want to use the connector, use the Connector Migration Tool to migrate jobs to use the connector.

This chapter describes the following for the Teradata Multiload stage:

- "Functionality of Teradata MultiLoad/TPump/FastExport Stages"
- "Building a MultiLoad or TPump Script"
- "Building a FastExport Script"
- "Writing Status Messages when Tracing Is Enabled"
- "Data Type Support"

Introduction to Teradata MultiLoad/TPump/FastExport Stage

IBM InfoSphere DataStage provides the ability to use the following Teradata utilities by generating the scripts that import or export data to or from a Teradata database.

- **MultiLoad** - Inserts, updates, deletes, or upserts rows in a Teradata table by using the MLOAD command-line utility. It can handle multiple SQL statements in a single operation. You can provide user-defined SQL for complex operations. This stage is best used for bulk updates, deletes, upserts, and complex interface operations.

- **TPump** - Inserts, updates, deletes, or upserts rows in a Teradata table using the TPUMP command-line utility. It can do concurrent updates on the same table. It can handle multiple SQL statements in a single operation. You can provide user-defined SQL for complex operations.

- **FastExport** - Exports data from a Teradata table by using the FEXP command-line utility. You can customize it to use user-defined SQL statements for extraction. This stage is best used for bulk extracts from Teradata.

Reference links have no meaning in the context of this stage and are not allowed.

**Input Links**

Input links utilize the MultiLoad utility to perform fast, high-volume maintenance functions on multiple tables and views of a Teradata database. Input links utilize the TPump utility to do concurrent updates on the same table. The Teradata Multiload/TPump/FastExport stage generates a MultiLoad script or a TPump script allowing you to run the load utility through IBM InfoSphere DataStage or invoke the load utility manually.
Output Links

Output links Utilize the FastExport utility to transfer large amounts of data quickly from tables and views of a Teradata database. Teradata MultiLoad/TPump/FastExport generates a FastExport script allowing you to run FastExport through IBM InfoSphere DataStage.

Installing the Teradata MultiLoad/TPump/FastExport Stage

Before installing the stage, consult Teradata documentation for any specific configuration requirements.

Configuring the Environment for Teradata Stages

About this task

Some Teradata stages require the setting of environment variables in order to work correctly on a UNIX platform. To add or change an environment variable, include any environment variables in the dsenv file. The following table identifies the specific environment variables that are required.

Table 15. Required Environment Variables for UNIX

<table>
<thead>
<tr>
<th>For...</th>
<th>Set the following environment variable on the IBM InfoSphere DataStage server machine...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teradata API stage</td>
<td>LD_LIBRARY_PATH</td>
</tr>
<tr>
<td></td>
<td>PATH</td>
</tr>
<tr>
<td>Teradata Load stage</td>
<td>LD_LIBRARY_PATH</td>
</tr>
<tr>
<td></td>
<td>PATH</td>
</tr>
<tr>
<td>Teradata MultiLoad/TPump/FastExport stage</td>
<td>LD_LIBRARY_PATH</td>
</tr>
<tr>
<td></td>
<td>PATH</td>
</tr>
</tbody>
</table>

The name of one particular environment variable, referred to as LD_LIBRARY_PATH above, differs depending on the platform. See the following to determine the correct name to use for your environment:

- If the platform is IBM AIX, use LIBPATH.
- If the platform is HP_UX, use SHLIB_PATH.
- If the platform is Linux, Solaris, or Tru64, use LD_LIBRARY_PATH.

Teradata Stages and the Parallel Canvas

Some connectivity stages were designed for server jobs but can run on the parallel canvas. The default for all stages is Sequential. "In Parallel" mean you can set it to run in parallel, but this is not the default. Use the following table to determine which connectivity server stages are available on the parallel canvas.
Table 16. Availability on the Parallel Canvas

<table>
<thead>
<tr>
<th>Connectivity Stage</th>
<th>Available on the Parallel Canvas in Windows</th>
<th>Available on the Parallel Canvas in UNIX</th>
<th>Used as a Source or a Target or for Processing</th>
<th>Runs Sequentially or In Parallel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teradata API</td>
<td>Yes</td>
<td>Source or Target</td>
<td>Source: Sequential; Target: In Parallel</td>
<td></td>
</tr>
<tr>
<td>Teradata Load</td>
<td>Yes</td>
<td>Target</td>
<td>Sequential</td>
<td></td>
</tr>
<tr>
<td>Teradata MultiLoad/TPump/FastExport</td>
<td>Yes</td>
<td>Source or Target</td>
<td>Sequential</td>
<td></td>
</tr>
<tr>
<td>Teradata connector</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Sequential and Parallel</td>
</tr>
</tbody>
</table>

Mapping String Data

The purpose of the NONE map on the server canvas is to turn off mapping of string data in any stage in which the map is set, i.e., to pass the data through verbatim. This feature is handled differently on the parallel canvas. When you define string data (char, varchar, etc.), there is an additional field in the stage's Columns grid called Extended. This can be set to blank or Unicode. If this option is set to blank, no mapping occurs (i.e. "NONE"); the map specified on the NLS tab is ignored. If this option is set to Unicode, the NLS map is applied. In order to read or write Japanese data, for example, set Extended to Unicode. When the job compiler detects this combination (char, varchar, etc. and Unicode), it generates the appropriate runtime code.

Functionality of Teradata MultiLoad/TPump/FastExport Stages

The Teradata MultiLoad/TPump/FastExport stage has the following functionality:
- Support for data files which exceed the 2 GB file size limit for 64-bit file systems.
- Generation, and optional automatic execution, of the Teradata commands:
  - To load or update a database with data from input links (MultiLoad or TPump).
  - To unload data from a database to output links (FastExport).
- Two load modes: manual or automatic.
- Load parameters to control the load process.
- NLS (National Language Support).

The following functionality is not supported:
- Compatibility with InfoSphere DataStage releases before 7.0
- Reference output links
- metadata importing
- Support for stored procedures
- Native data browsing
- Support for reject row handling
Building a MultiLoad or TPump Script

About this task

You need to specify the stage properties when you build a MultiLoad or TPump script. To do this:

Procedure
1. Create an IBM InfoSphere DataStage job.
2. Define the properties used by InfoSphere DataStage and QualityStage Designer to construct the script.
3. Compile and run the job.

Results

Each task is described in more detail in the following sections.

Creating a InfoSphere DataStage Job to Load Data

Procedure
1. Create a job using the Designer client.
2. Choose TDMLoad from the Insert menu or select the icon from the Designer client job palette.
3. Add an input stage and add a link.

Defining Properties to Load Data

Right click the TDMLoad icon and select Properties or choose Properties from the Edit menu. The TDMLoad Stage dialog box opens. This dialog box has two pages:
- **Stage**: Displays the name of the stage you are editing. The **General** tab defines the Teradata data target and logon information.
- **Input**: Specifies the information necessary to generate a MultiLoad or TPump script. This page also specifies the associated column definitions.

About the Stage Page

Supply information on the **General** tab on the Stage page to identify the target of the import.

The **General** tab has the following fields:
- **Server**: The name of a Teradata Director Program (TDP). Optional.
- **Username**: A name that identifies the user. The user must have the necessary privileges to write to the database. Required.
- **Password**: The password associated with the user name. Required.
- **Account**: The account associated with the user name. Optional.
- **Database**: The name of the database to be loaded or updated. Optional.
- **Client character set**: The Teradata client character set to use when connecting to the Teradata server. The default value is **Default**.
- **Description**: A description of the stage. Optional.

About The Input Page

The Input page has an **Input name** field, a **Columns**... button, and General, MultiLoad/TPump, TPump, Limit Settings, Script, and Columns tabs.
Input name. The name of the input link. Choose the link you want to edit from the Input name list. This list displays all the input links to the TDMLoad stage.

Click Columns... to display a brief list of the columns designated on the input link. As you enter detailed metadata on the Columns tab, you can leave this list displayed.

General Tab of the Input Page
Use the General tab to specify details about how data will be loaded.

The General tab has the following fields:

- **Overwrite Existing Script File.** The disposition of the current script.
  - If you want to replace the current script each time the job is compiled, select Overwrite Existing Script File. This is the default.
  - If you want to make and save modifications directly to the generated script, clear Overwrite Existing Script File.

- **Use Named Pipe.** The procedure for loading data.
  - If you want to stream rows of data to the load utility (MultiLoad or TPump) via a named pipe, select Use Named Pipe. This is the default.
  - If you want to write all rows to a data file and then start the load utility that loads from the data file, clear Use Named Pipe.

- **Use Large File AXSMOD (UNIX ONLY).** An indicator allowing the processing of files larger than 2 gigabytes. This feature is available only on UNIX platforms and is an extension to MultiLoad and TPump. To use this feature, you must have AXSMOD if_AXSMOD.so (or .sl) installed on the server. This file is provided by Teradata. This property is available only if Use Named Pipe is cleared. The default is Use Large File AXSMOD selected.

- **Overwrite Existing Data File.** The procedure for handling an existing data file. This property is available only if Use Named Pipe is cleared.
  - If you want to replace the current data file each time the job is run, select Overwrite Existing Data File. This is the default.
  - If you want to ignore the input rows and start the load utility, which must load from an existing data file, clear Overwrite Existing Data File.

- **Delete Data File After Load.** The disposition of a data file after a load. This property is available only if Use Named Pipe is cleared.
  - If you want to delete the data file after the data is loaded successfully, select Delete Data File After Load. This is the default.
  - If you want to leave the data file intact after loading completes, clear Delete Data File After Load.

- **Data Format.** The format in which rows are written to the data file.
  - If FASTLOAD is selected, rows are written in the Teradata RDBMS standard format. This is the default.
  - If VARTEXT is selected, rows are written in variable-length text record format with each field separated by a delimiter character.

- **Delimiter.** The delimiter character that separates fields in variable-length text records. This property is available only if Data Format is VARTEXT. The default delimiter is the pipe character (|).

- **Description.** A description of the script. Optional.

MultiLoad/TPump Tab
Use the MultiLoad/TPump tab to provide general information about the script to be generated.
The MultiLoad/TPump tab contains the following fields:

- **Table.** The name of the table to be loaded. The name is used in the BEGIN MLOAD command in the generated script. Required.

- **Load Utility.** Two mutually exclusive option buttons used to specify either the MultiLoad utility or the TPump utility as the load method. The default is MultiLoad selected.

- **Report File.** The name of the report file created by the load utility at execution time. The default name is table.txt, where table is the name supplied in the Table text box, or database_table.txt, if you supply a database name in the Database text box.

- **Control File.** The name of the generated script. The default name is table.fl, where table is the name supplied in the Table text box, or database_table.fl, if you supply a database name in the Database text box.

- **Data File.** The name of the data file if you clear Use Named Pipe or if the Load Method is Manual. The default name is table.dat, where table is the name supplied in the Table text box, or database_table.dat, if you supply a database name in the Database text box.

- **Load Method.** The timing of the execution of the script.
  - If you select Invoke Load Utility, the load utility is executed automatically when the job is run. The stage creates a named pipe to transmit data to the load utility, and then it starts the load process. The load process has 12 minutes (720 seconds) to reach the phase in which the utility opens the named pipe to read the data. This is called the Acquisition Phase. If the load utility fails to reach Acquisition Phase and open the named pipe before the time has expired, the stage aborts and stops the process. If the default timeout is insufficient, you can override the default by doing one of the following:
    - Define the following environment variable in the IBM InfoSphere DataStage and QualityStage Administrator.
      \[
      \text{DS_TDM_PIPE_OPEN_TIMEOUT} \ n
      \]
      \(n\) is the number of seconds the stage waits before aborting.
    - Use the Tenacity option on the Limit Settings tab.
  - If you select Manual, the data to be imported is stored as a .dat file. You can then execute the script, which points to the .dat file, and load the data independent of the InfoSphere DataStage job. The default is Invoke Load Utility selected.

- **Load Type.** The type of load activity taking place when the load utility selected in Load Utility executes. The options are:
  - **Insert.** Instructs the load utility to insert all acceptable rows.
  - **Update.** Instructs the load utility to update any rows for which there is a key match.
  - **Delete.** Instructs the load utility to delete any rows for which there is a key match.
  - **Upsert.** Instructs the load utility to update any rows for which there is a key match and insert any acceptable rows for which there is no key match.
  - **Custom.** Generates a custom script to load data.
  
  The default is Insert selected.

- **Error Table 1.** The name of an error table used by the load utility when it detects errors during execution of the script. See your Teradata MultiLoad or TPump documentation.

- **Error Table 2.** The name of a second error table used by MultiLoad when it detects errors during execution of the script. TPump does not use a second error
• **Log Table.** The name of the log table used by the load utility if the script includes a .log table command.

• **Work Tables.** The names of the special unhashed tables used by the MultiLoad utility when executing both import and delete tasks. TPump does not use work tables, and this text box is unavailable if you select the TPump option button. See your Teradata MultiLoad documentation.

• **Output Files Path.** The name of the path to be used for the Report File, the Control File, and the Data File. Use the Browse button to facilitate identifying the path.

**TPump Tab**

This tab contains options that apply only to the TPump utility. These options are not available if you select the MultiLoad option button.

The following properties are included on the TPump tab:

• **Latency.** The maximum number of seconds that a record resides in the TPump buffers before the buffers are flushed. The default value of 0 indicates the Teradata default should be used.

• **Pack.** The number of statements to pack into a multiple-statement request. The default value of 0 indicates the Teradata default should be used.

• **Rate.** The number of statements per minute to be sent to the Teradata database. The default value of 0 indicates the Teradata default should be used.

• **Serialize.** The way multiple operations on a given row are guaranteed to occur. If On is selected, the multiple operations occur serially. The default is Off selected.

• **Robust.** The restart logic to be used. If Off is selected, TPump uses simpler but less reliable restart logic. The default is On selected.

• **Job Name.** A unique identifier assigned to the TPump environment variable SYSJOBNAME. TPump truncates the identifier to 16 characters. If not provided, TPump uses the Teradata default.

• **Macro Database.** The name of the database that is to contain any macros built or used by TPump. If not provided, TPump uses the Teradata default.

**Limit Settings Tab**

Limit Settings are used in the BEGIN MLOAD or BEGIN LOAD command and correspond directly to options in the command.

The following properties are included on the Limit Settings tab:

• **Limit Settings.** The settings include:
  – Sess Max
  – Sess Min
  – Error Limit
  – Checkpoint
  – Start Row
  – End Row
  – Sleep
  – Tenacity

  See your Teradata MultiLoad or TPump documentation for descriptions of these settings.
If you select TPump on the MultiLoad/TPump tab (see “MultiLoad/TPump Tab” on page 115), you must specify a value in Sess Max greater than 0.

- Other BEGIN (M)Load Clauses. Additional clauses for the .BEGIN LOAD or .BEGIN MLOAD statement that are not covered by the other properties.

Script Tab
Use the Script tab to build your own script to run the load utility. This capability allows you to take advantage of the power of MultiLoad or TPump and create a script having some combination of the Update, Insert, Delete, and Upsert load types. The Script tab consists of four tabs: Generated, Custom, Before, and After.

Generated Tab
The Generated tab displays the generated MultiLoad or TPump script.

Custom Tab
Build your custom script in the Custom Script text box.

Start with the DML LABEL command and end with the IMPORT command. You can customize statements between the .BEGIN LOAD and .END LOAD statements (TPump) or between the .BEGIN MLOAD and .END MLOAD statements (MultiLoad). You must also supply column names and values for each option you choose. See your Teradata MultiLoad or TPump documentation for additional information.

The stage provides assistance in creating a custom script. Click Generate Sample to populate the text box with a sample script based on the metadata supplied to the stage. The generated sample will not run unless it is modified. At a minimum you must activate the section beginning .import:

- If you are working in a UNIX environment, change the information immediately to the right of /*UNIX*/ by removing the "/" and "/.
  For example, change the line
  
  /*UNIX*/ /*DSLINK11*/
  
  to
  
  /*UNIX*/ DSLINK11

- If you are working in a Windows environment, change the information immediately to the right of /*NT*/ by removing the "/" and "/.
  For example, change the line
  
  /*NT*/ /*\\.pipe\tdmpipe_DSLINK11 AXSMOD np_AXSMOD.dll*/
  
  to
  
  /*NT*/ \\
  .pipe\tdmpipe_DSLINK11 AXSMOD np_AXSMOD.dll

Before and After Tabs
The Before and After tabs allow additional customizing. They apply to both MultiLoad and TPump and to every Load Type: Insert, Update, Delete, Upsert, and Custom.

- Before tab. Insert custom statements before the .BEGIN LOAD or .BEGIN MLOAD statement in the generated script using the text box.
- After tab. Insert custom statements after the .END LOAD or .END MLOAD statement in the generated script using the text box.

A statement can span multiple lines. Each statement must begin on a new line and end with a semicolon. MultiLoad or TPump commands that are not SQL statements must begin with a period.
If **Drop error table(s)** is selected, the generated script drops the error tables before loading begins. If **Drop error table(s)** is not selected, which is the default, the stage assumes one of the following:

- There are no error tables to drop. Therefore, the stage does not put statements to drop the error tables in the generated script. If either the log table or error tables preexist, MultiLoad or TPump fails.
- You are attempting a restart. Therefore the log table and error tables must preexist from a previous run.

In order to drop the error table, do the following:

- If you select **TPump** on the MultiLoad/TPump tab, provide a name in either **Error Table 1** on the MultiLoad/TPump tab (see “MultiLoad/TPump Tab” on page 115) or in **Job Name** on the TPump tab (see “TPump tab” on page 117). If you do not provide a name in **Error Table 1**, TPump uses **JobName_ET** as the error table name.

- If you select **MultiLoad** on the MultiLoad/TPump tab, provide names in **Error Table 1** and **Error Table 2** (see “MultiLoad/TPump Tab” on page 115). If you do not provide names in **Error Table 1** and **Error Table 2**, MultiLoad uses **ET_TableName** and **UV_TableName** as the error table names (see “MultiLoad/TPump Tab” on page 115).

The following property is included on the **Before** and **After** tabs:

- **Treat SQL errors as non-fatal.** The conditions under which processing stops with regard to Before and After SQL statements.
  - If **Treat SQL errors as non-fatal** is selected, errors caused by Before SQL statements or After SQL statements are ignored, and processing continues. This is the default.
  - If **Treat SQL errors as non-fatal** is not selected, errors from SQL statements are treated as fatal to the job.

The following property is included on the **After** tab:

- **Fail DataStage Job on Error Table rows.** During the load process, MultiLoad populates two error tables if rows contain errors. Similarly, TPump populates one error table if rows contain errors. See “MultiLoad/TPump Tab” on page 115. If you select **Fail DataStage Job on Error Table rows**, the job aborts if the load utility inserts any rows into those tables. The default is **Fail DataStage Job on Error Table rows** selected.

**Columns Tab of the Input Page**

This tab contains the metadata for the data to be imported to the Teradata database.

You can manually enter the metadata in the grid or load it from IBM InfoSphere DataStage repository. The columns listed determine what is imported to the Teradata database.

Click **Save** to save the metadata to InfoSphere DataStage repository. Click **Load** to load the metadata from the repository.

**Compiling and Running the Job**

Complete the definition of the other stages in your job design according to normal IBM InfoSphere DataStage procedures. Compile and run the job.
Enhancing Performance of a MultiLoad Script

About this task

If you experience unsatisfactory load performance when running a MultiLoad script, consider reading data directly from a sequential file source. This will eliminate the need for the data to be processed by IBM InfoSphere DataStage internals. To accomplish this:

Procedure

1. In TDMLoad on the General tab of the Input page,
   a. Clear Use Named Pipe. This causes MultiLoad to read from a sequential file instead of a pipe.
   b. Clear Use Large File AXSMOD.
   c. Clear Overwrite Existing Data File. This causes MultiLoad to read from an existing file instead of letting the stage create it.
   d. Change Data Format to VARTEXT. This assumes your input sequential file is a text-delimited file and not a binary file in FastLoad format.

   Note: When you use VARTEXT, MultiLoad expects that fields are not enclosed in quotation marks.
   e. Set Delimiter to the field delimiter character you are using in the input sequential file.

2. In TDMLoad on the MultiLoad/TPump tab of the Input page, enter the name of the sequential file you want to load in Data File.

3. Because TDMLoad requires an input link, precede TDMLoad on the canvas with a dummy stage that passes no data to TDMLoad. This stage can be a Sequential File, a Transformer or Row Generator (on the parallel canvas).
   • If the stage is a Sequential File, make sure the file is empty. Specifying an empty file insures the best performance.
   • If the stage is a Transformer, put a constraint on the output link, such as @FALSE.
   • If the stage is a Row Generator, specify 0 rows in Number of Records.

Results

When TDMLoad executes, it achieves optimal performance.

Building a FastExport Script

About this task

You need to specify the stage properties when you build a FastExport script. To do this:

Procedure

1. Create an IBM InfoSphere DataStage job.
2. Define the properties used by the Designer client to construct the script.
3. Compile and run the job.

Results

Each task is described in more detail in the following sections.
Creating an InfoSphere DataStage Job to Export Data

To create an IBM InfoSphere DataStage job, do the following steps:

**Procedure**
1. Create a job using the Designer client.
2. Choose **TDMLoad** from the **Insert** menu or select the icon from the Designer client job palette.
3. Add an output stage and add a link.

**Defining Properties to Export Data**
Right click the **TDMLoad** icon and select **Properties** or choose **Properties** from the **Edit** menu. The TDMLoad Stage dialog box opens. This dialog box has two pages:

- **Stage.** Displays the name of the stage you are editing. The **General** tab defines the Teradata data source and logon information.
- **Output.** Specifies the information necessary to generate a FastExport script. This page also specifies the associated column definitions.

**About the Stage Page**
Supply information on the **General** tab on the Stage page to identify the target of the import. The **General** tab supports the following properties:

- **Server.** The name of a Teradata Director Program (TDP). Optional.
- **Username.** A name that identifies the user. The user must have the necessary privileges to read from the database. Required.
- **Password.** The password associated with the user name. Required.
- **Account.** The account associated with the user name. Optional.
- **Database.** The name of the database from which data is to be exported. Optional.
- **Description.** A description of the stage. Optional.

**About the Output Page**
The Output page has an **Output name** field, a **Columns...** button, and **General, FastExport, Selection, SQL, and Columns** tabs.

- **Output name.** The name of the output link. Choose the link you want to edit from the **Output name** list. This list displays all the output links from the TDMLoad stage.
- **Click Columns...** to display a brief list of the columns designated on the output link. As you enter detailed metadata on the **Columns** tab, you can leave this list displayed.

**General Tab of the Output Page**
Use the **General** tab to specify details about the export.

The following properties are included on the **General** tab:

- **Overwrite Existing Script File.** The disposition of the current script.
  - If you want to replace the current script each time the job is compiled, select **Overwrite Existing Script File.** This is the default.
  - If you want to make and save modifications directly to the generated script, clear **Overwrite Existing Script File.**
- **Use Named Pipe.** The procedure for extracting data.
– If you want to stream rows of data from the FastExport utility via a named pipe, select **Use Named Pipe.** This is the default.
– If you want FastExport to write all rows to a data file and then have the stage read the data file and send rows to the output link, clear **Use Named Pipe.**

**Use Large File AXSMOD of FastExport (UNIX ONLY).** An indicator allowing the processing of files larger than 2 gigabytes. This feature is available on UNIX platforms only and is an extension to FastExport. To use this FastExport feature, you must have `AXSMOD lf_AXSMOD.so` (or `.sl`) installed on the server. This file is provided by Teradata. The default is **Use Large File AXSMOD of FastExport** selected.

**Note:** If you are using Teradata client software TTU 7.0, do not select **Use Large File AXSMOD of FastExport.** Large File Access Module is obsolete with this release of Teradata client software. Failure to clear **Use Large File AXSMOD of FastExport** results in an error.

**Overwrite Existing Data File.** The procedure for handling an existing data file. This property is available only if **Use Named Pipe** is cleared.
– If you want to replace the current data file each time the job is run, select **Overwrite Existing Data File.** This is the default.
– If you want to read from an existing data file and not run FastExport, clear **Overwrite Existing Data File.**

**Delete Data File After Export.** The disposition of a data file after an export. This property is available only if **Use Named Pipe** is cleared.
– If you want to delete the data file after the data has been read successfully, select **Delete Data File After Export.** This is the default.
– If you want to leave the data file intact after all rows have been read, clear **Delete Data File After Export.**

**Select using Generated Query.** The method of building a FastExport script by using an SQL query generated from selection criteria. The default is **Generated Query** selected.

**Select using User-defined SQL Query.** The method of building a FastExport script by using a manually created SQL query. The default is **User-defined SQL Query** cleared.

**Description.** A description of the script. Optional.

**FastExport Tab**

Use the **FastExport** tab to provide general information about the script to be generated.

The following properties are included on the **FastExport** tab:

**Table.** The name of the table to be exported. The name is used in the EXPORT command in the generated script. Required.

**Report File.** The name of the report file created by FastExport at execution time. The default name is `table.txt`, where `table` is the name supplied in the **Table** text box.

**Control File.** The name of the generated script. The default name is `table.fl`, where `table` is the name supplied in the **Table** text box.

**Log Table.** The name of the log table used by FastExport if the script includes a .log table command.

**Output Files Path.** The name of the path to be used for the Report File and the Control File. Use the **Browse** button to facilitate identifying the path.
• **Limit Settings.** Limit Settings are used in the BEGIN EXPORT command and correspond directly to options in the command. The settings include:
  - Sess Max
  - Sess Min
  - Sleep
  - Tenacity
  
  See your Teradata FastExport documentation.

• **Data File.** The name of the data file if you clear **Use Named Pipe**. The default name is table.dat, where table is the name supplied in the **Table** text box, or database_table.dat, if you supply a database name in the **Database** text box.

**Selection Tab**
Use this tab to specify the selection criteria used to generate the SQL query for the FastExport script. The following properties are included on the **Selection** tab:

• **WHERE clause.** An SQL WHERE clause specifying criteria the data must meet before being selected.
• **Other clauses.** An optional GROUP BY, HAVING, or ORDER BY clause to sort, summarize, and aggregate data.

**SQL Tab**
Use this tab to construct the complete SQL Query for the generated FastExport script.

The SQL tab is active only if you choose **User-defined SQL Query** on the **General** tab.

**Columns Tab of the Output Page**
This tab contains the metadata for the data to be exported from the Teradata database.

You can manually enter the metadata in the grid or load it from IBM InfoSphere DataStage repository. The columns listed determine what is exported from the Teradata database. The **Column name** must be the same as the physical name or alias in the Teradata database or you must specify a **Derivation** that is the same as the physical name or alias in the Teradata database.

Click **Save** to save the metadata to InfoSphere DataStage repository. Click **Load** to load the metadata from the repository.

**Compiling and Running the Job**
Complete the definition of the other stages in your job design according to normal IBM InfoSphere DataStage procedures. Compile and run the job.

**Writing Status Messages when Tracing Is Enabled**
To aid with troubleshooting, you can write status messages to IBM InfoSphere DataStage log if tracing is enabled.

• **To enable tracing when executing a job immediately.** In the InfoSphere DataStage and QualityStage Director, on the **Tracing** tab in the Job Run Options dialog box, select the active stages that connect to Teradata MultiLoad. Then select **Subroutine calls**.
• **To enable tracing in all jobs.** In the Administrator client, define the following environment variable and give it a value of 1.
To return to the default of tracing only when set from the Job Run Options dialog box, remove the variable or set it to 0.

In all cases, when tracing is turned off, the stage writes the following status messages to the InfoSphere DataStage log where UtilityName is either MultiLoad, TPump, or FastExport:
- JobName..StageName: UtilityName process ProcessNumber has started
- JobName..StageName: UtilityName has completed. Report file: ReportFileName

When tracing is turned on, the stage writes the following MultiLoad or TPump status messages to the InfoSphere DataStage log where LoadUtilityName is either MultiLoad or TPump:
- JobName..StageName: Initializing link InputLinkName
- JobName..StageName: Getting properties for link InputLinkName
- JobName..StageName: Validating columns for link InputLinkName
- JobName..StageName: Creating files for link InputLinkName
- JobName..StageName: Starting process mload < ControlFileName > ReportFileName 2>&1
  or
- JobName..StageName: Starting process tpump < ControlFileName > ReportFileName 2>&1
- JobName..StageName: LoadUtilityName process ProcessNumber has started
- JobName..StageName: Opening file DataFileName
- JobName..StageName: Started with the following parameters: Server: ServerName Username: UserName Password: Sorry its Encrypted Account: AccountName Database: DatabaseName Control File: ControlFileName Table: TableName Error Table 1: ErrorTable1Name Error Limit: 0 Sessions: 0 Row Start: 0 Row End: 0 Run LoadUtilityName: Invoke LoadUtilityName Drop Table:
  Dir path: Base name:
- JobName..StageName: Closing link InputLinkName
- JobName..StageName: Closing file DataFileName
- JobName..StageName: Waiting for process ProcessNumber to complete
- JobName..StageName: Getting return code for process LoadUtilityName
- JobName..StageName: Deleting file DataFileName
- JobName..StageName: LoadUtilityName has completed.
- JobName..StageName: Freeing resources for link InputLinkName

When tracing is turned on, the stage writes the following FastExport status messages to the InfoSphere DataStage log:
- JobName..StageName: Initializing link OutputLinkName
- JobName..StageName: Getting properties for link OutputLinkName
- JobName..StageName: Validating columns for link OutputLinkName
- JobName..StageName: Creating files for link OutputLinkName
- JobName..StageName: Starting process fexp -r ".run file ControlFileName;" > /dev/null 2>&1
- JobName..StageName: FastExport process ProcessNumber has started
- JobName..StageName: Opening file DataFileName
- JobName..StageName: Closing link OutputLinkName
- JobName..StageName: Waiting for process ProcessNumber to complete
- JobName..StageName: Getting return code for process fexp
- JobName..StageName: FastExport has completed. Report file: ReportFileName
- JobName..StageName: Freeing resources for link OutputLinkName
Data Type Support

The following table documents the support for Teradata data types:

Table 17. Support for Teradata data types

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unknown</td>
<td>Unsupported</td>
</tr>
<tr>
<td>BigInt</td>
<td>Unsupported</td>
</tr>
<tr>
<td>Binary</td>
<td>Supported</td>
</tr>
<tr>
<td>Bit</td>
<td>Supported</td>
</tr>
<tr>
<td>Char</td>
<td>Supported</td>
</tr>
<tr>
<td>Date</td>
<td>Supported</td>
</tr>
<tr>
<td>Decimal</td>
<td>Supported</td>
</tr>
<tr>
<td>Double</td>
<td>Supported</td>
</tr>
<tr>
<td>Float</td>
<td>Supported</td>
</tr>
<tr>
<td>Integer</td>
<td>Supported</td>
</tr>
<tr>
<td>LongNVarChar</td>
<td>Unsupported</td>
</tr>
<tr>
<td>LongVarBinary</td>
<td>Unsupported</td>
</tr>
<tr>
<td>LongVarChar</td>
<td>Unsupported</td>
</tr>
<tr>
<td>NChar</td>
<td>Unsupported</td>
</tr>
<tr>
<td>Numeric</td>
<td>Supported</td>
</tr>
<tr>
<td>NVarChar</td>
<td>Unsupported</td>
</tr>
<tr>
<td>Real</td>
<td>Supported</td>
</tr>
<tr>
<td>SmallInt</td>
<td>Supported</td>
</tr>
<tr>
<td>Time</td>
<td>Supported</td>
</tr>
<tr>
<td>Timestamp</td>
<td>Supported</td>
</tr>
<tr>
<td>TinyInt</td>
<td>Supported</td>
</tr>
<tr>
<td>VarBinary</td>
<td>Supported</td>
</tr>
<tr>
<td>VarChar</td>
<td>Supported</td>
</tr>
</tbody>
</table>

The Teradata MultiLoad/TPump/FastExport stage assumes a Char or VarChar column is Unicode if the value specified in Display on the Columns tab is double the value specified in Length. Therefore the Generated tab on the Script tab shows the field declared using the value specified in Display instead of in Length.
Chapter 8. Environment variables: Teradata connector

The Teradata Connector stage uses these environment variables.

**CC_GUARDIUM_EVENTS**

Set this environment variable to specify whether connectors report the InfoSphere DataStage context information to the InfoSphere Guardium Database Activity monitor.

When the value of this environment variable is set, the connectors report the InfoSphere DataStage context information such as host, project, job names, stage name and node ID that the stage is running on to the InfoSphere Guardium Database Activity monitor. When this environment variable is defined and set to any value, the connectors report context information to the Guardium server after the initial connection is established.

When this environment variable is undefined, the connectors do not attempt to report context information to Guardium servers. The setting of this environment variable applies to all database connectors in the job.

**CC_IGNORE_TIME_LENGTH_AND_SCALE**

Set this environment variable to change the behavior of the connector on the parallel canvas.

When this environment variable is set to 1, the connector running with the parallel engine ignores the specified length and scale for the timestamp column. For example, when the value of this environment variable is not set and if the length of the timestamp column is 26 and the scale is 6, the connector on the parallel canvas considers that the timestamp has a microsecond resolution. When the value of this environment variable is set to 1, the connector on the parallel canvas does not consider that the timestamp has a microsecond resolution unless the microseconds extended property is set even if the length of the timestamp column is 26 and the scale is 6.

**CC_MSG_LEVEL**

Set this environment variable to specify the minimum severity of the messages that the connector reports in the log file.

At the default value of 3, informational messages and messages of a higher severity are reported to the log file.

The following list contains the valid values:

- 1 - Trace
- 2 - Debug
- 3 - Informational
- 4 - Warning
- 5 - Error
- 6 - Fatal
**CC_SE_TIMESTAMP_FF**

Set this environment variable to specify whether decimal point and fractional digits are included in the timestamp values, when the connector runs in server jobs.

When the environment variable is set to a value other than NONE, MICROSECONDS or SCALE, the behavior is the same as if the environment variable was not set. The environment variable values are case sensitive. When the environment variable is not set, the timestamp values that are produced by the job include a trailing decimal point and six fractional digits.

You can set the environment variable to the following values:

- **NONE**
  - The trailing decimal point and the fractional digits are both omitted.

- **MICROSECONDS**
  - The trailing decimal point and six fractional digits are included.

- **SCALE**
  - The trailing decimal point and $S$ fractional digits are included, where $S$ represents the value of the Scale attribute in the timestamp column definition. When the Scale attribute value is not defined for the column, the Scale attribute value of zero is assumed.

**CC_USE_EXTERNAL_SCHEMA_ON_MISMATCH**

Set this environment variable to use an external schema rather than a design schema when the schemas do not match.

This schema is used for schema reconciliation. When the value of this environment variable is set, the behavior remains the same and is not changed from the old version.

**CC_TERA_DEBUG**

Set this environment variable, in addition to the **CC_MSG_LEVEL** variable, to specify the level of logging and tracing in the Teradata connector.

If the **CC_MSG_LEVEL** environment variable is set to 1 or 2, you can specify the level of logging and tracing by setting the **CC_TERA_DEBUG** environment variable to one of the following values:

1. Connector records the Teradata Call-Level interface structures after each call to the Teradata client.
2. Connector tests some of the metadata import methods.
3. Connector records the Teradata Call-Level interface structures and tests the metadata import methods.
4. Connector enables tracing for the Teradata Parallel Transporter and writes the output to a file name beginning with TD_TRACE_OUTPUT in the project directory.
5. Connector records the trace output of the Teradata Parallel Transporter.

When this environment variable is undefined, the connector does not provide additional logging or tracing.
**CC_TERA_FAIL_ON_DBS_CRASH**

Set this environment variable to specify whether the connector ends the job when a Teradata database crashes or restarts.

When this environment variable is set to 1 or any other nonzero value, the connector ends the job when the database crashes or restarts. When this environment variable is set to 0 or is undefined, the connector does not end the job.

**CC_TERA_MAX_DECIMAL_RETURN**

Set this environment variable to specify the number of decimal places that are returned by the Teradata server for decimal values.

When this environment variable is set to an integer value, the connector uses that integer value as the number of decimal places to return by the Teradata server.

When this environment variable is undefined, the connector sets the maximum number of decimal places to 38.

**CC_TERA_OPTIMIZE_CONNECTIONS**

Set this environment variable to disconnect the conductor node's SQL sessions while the player nodes are processing data.

When this environment variable is set to 1 or any other nonzero value, the connector saves resources by disconnecting conductor sessions while the player nodes are processing data. When this environment variable is set to 0 or is undefined, the connector keeps the conductor node's connected.

**CC_TERA_RECONNECT_FLAG**

Set this environment variable to specify whether to use the legacy reconnect features.

When this environment variable is set to 1 or any other nonzero value, the connector attempts to reconnect any sessions that are unexpectedly disconnected and return Teradata RDBMS codes 8055 or 8018 to the job log. When this environment variable is set to 0 or is undefined, the connector ends the job if a session is unexpectedly disconnected. This environment variable overrides the reconnect features that are defined in the **Reconnect** and **Disconnect** properties.

**CC_TERA_TMSM_LOG_EVENT**

Set this environment variable to specify whether Teradata Multi-System Manager events are logged in the InfoSphere DataStage Director log.

When this environment variable is set to 1, information about the Teradata Multi-System Manager events in a job are logged in the Director log. When this environment variable is undefined or set to any value except 1, the connector does not log Teradata Multi-System Manager events in the Director log. The connector follows the setting of the **CC_TERA_TMSM_LOG_EVENT** variable regardless of the setting for the **TMSM event options** property.
**CC_TERA_USE_ISO88591_ASCII_MAP**

Set this environment variable to override the values for the **Client character set** property and the **Maximum bytes per character** property.

This environment variable also sets the effective encoding of the connector stage to ISO-8859-1 and is also used for jobs with NLSMap set to ISO-8859-1.

When this environment variable is set to any value, the connector does the following actions:
- Sets the **Client character set** property to **ASCII**
- Sets the **Maximum bytes per character** property to **1**
- Sets the effective encoding of the connector to ISO-8859-1 (Latin -1)
- Ignores the values that are defined for the **Client character set** and **Maximum bytes per character** properties in the Teradata connector stage.

The **CC_TERA_USE_ISO88591_ASCII_MAP** environment variable is used only when the **Automap character encoding** property is set to YES. When this environment value is not defined, the connector uses the values of the **Client character set** and **Maximum bytes per character** properties and the NLS map that is defined in the job.

**CC_TRUNCATE_NSTRING_WITH_NULL**

Set this environment variable to truncate string data that includes the string 0x00.

When the value of this environment variable is set and when the input data contains a null character, the input data is truncated with 0x00 and the rest of the string is dropped.

**DS_NO_PROCESS_METADATA**

Set this environment variable to specify whether metadata queries of the MetaStage_Loc_Info table are executed.

When this environment variable is set to 1, or any other nonzero value, the connector does not query the MetaStage_Loc_Info table when the job runs. When this environment variable is set to 0, the connector queries the MetaStage_Loc_Info table.
Chapter 9. Building SQL statements

Use the graphical interface of SQL builder to construct SQL statements that run against databases.

You can construct the following types of SQL statements.

<table>
<thead>
<tr>
<th>SQL statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT</td>
<td>Selects rows of data from a database table. The query can perform joins between multiple tables and aggregations of values in columns.</td>
</tr>
<tr>
<td>INSERT</td>
<td>Inserts rows in a database table.</td>
</tr>
<tr>
<td>UPDATE</td>
<td>Updates existing rows in a database table.</td>
</tr>
<tr>
<td>DELETE</td>
<td>Deletes rows from a database table.</td>
</tr>
</tbody>
</table>

You can use the SQL from various connectivity stages that IBM InfoSphere DataStage supports.

Different databases have slightly different SQL syntax (particularly when it comes to more complex operations such as joins). The exact form of the SQL statements that the SQL builder produces depends on which stage you invoke it from.

You do not have to be an SQL expert to use the SQL builder, but it helps to have some familiarity with the basic structure of SQL statements in this documentation.

Starting SQL builder from a stage editor

You reach the SQL builder through the stage editors. Where the stage type supports the builder you will see a Build SQL button. Click this to open the SQL builder. For some stages you will have to select an appropriate access method for the button to be visible. See the documentation for the individual stage types for details.

The SQL builder is available to help you build select statements where you are using a stage to read a database (that is, a stage with an output link).

The SQL builder is available to help you build insert, update, and delete statements where you are using the stage to write to database (that is, a stage with an input link).

Starting SQL builder

Use the graphical interface of SQL builder to construct SQL queries that run against federated databases.

Procedure

1. In the Reference Provider pane, click Browse. The Browse Providers dialog box opens.
2. In the **Select a Reference Provider** type list, select **Federation Server**. In the Select a Federated Datasource tree, the list of database aliases opens.

3. Click a database alias. The list of schemas opens as nodes beneath each database alias.

4. In the **SQL Type** list, select the type of SQL query that you want to construct.

5. Click the **SQL builder** button. The SQL Builder - DB2 / UDB 8.2 window opens. In the Select Tables pane, the database alias appears as a node.

---

### Building SELECT statements

Build SELECT statements to query database tables and views.

**Procedure**

1. Click the **Selection** tab.

2. Drag any tables you want to include in your query from the repository tree to the canvas. You can drag multiple tables onto the canvas to enable you to specify complex queries such as joins. You must have previously placed the table definitions in the IBM InfoSphere DataStage repository. The easiest way to do this is to import the definitions directly from your relational database.

3. Specify the columns that you want to select from the table or tables on the column selection grid.

4. If you want to refine the selection you are performing, choose a predicate from the **Predicate** list in the filter panel. Then use the expression editor to specify the actual filter (the fields displayed depend on the predicate you choose). For example, use the Comparison predicate to specify that a column should match a particular value, or the Between predicate to specify that a column falls within a particular range. The filter appears as a WHERE clause in the finished query.

5. Click the **Add** button in the filter panel. The filter that you specify appears in the filter expression panel and is added to the SQL statement that you are building.

6. If you are joining multiple tables, and the automatic joins inserted by the SQL builder are not what is required, manually alter the joins.

7. If you want to group your results according to the values in certain columns, select the Group page. Select the Grouping check box in the column grouping and aggregation grid for the column or columns that you want to group the results by.

8. If you want to aggregate the values in the columns, you should also select the Group page. Select the aggregation that you want to perform on a column from the **Aggregation** drop-down list in the column grouping and aggregation grid.

9. Click on the **Sql** tab to view the finished query, and to resolve the columns generated by the SQL statement with the columns loaded on the stage (if necessary).

---

### Building INSERT statements

Build INSERT statements to insert rows in a database table.

**Procedure**

1. Click the **Insert** tab.

2. Drag the table you want to insert rows into from the repository tree to the canvas. You must have previously placed the table definitions in the IBM
InfoSphere DataStage repository. The easiest way to do this is to import the definitions directly from your relational database.

3. Specify the columns that you want to insert on the column selection grid. You can drag selected columns from the table, double-click a column, or drag all columns.

4. For each column in the column selection grid, specify how values are derived. You can type a value or select a derivation method from the drop-down list.
   - **Job Parameters.** The Parameter dialog box appears. Select from the job parameters that are defined for this job.
   - **Lookup Columns.** The Lookup Columns dialog box appears. Select a column from the input columns to the stage that you are using the SQL builder in.
   - **Expression Editor.** The Expression Editor opens. Build an expression that derives the value.

5. Click on the Sql tab to view the finished query.

---

### Building UPDATE statements

Build UPDATE statements to update existing rows in a database table.

**Procedure**

1. Click the **Update** tab.
2. Drag the table whose rows you want to update from the repository tree to the canvas. You must have previously placed the table definitions in the IBM InfoSphere DataStage repository. The easiest way to do this is to import the definitions directly from your relational database.
3. Specify the columns that you want to update on the column selection grid. You can drag selected columns from the table, double-click a column, or drag all columns.
4. For each column in the column selection grid, specify how values are derived. You can type a value or select a derivation method from the drop-down list. Enclose strings in single quotation marks.
   - **Job Parameters.** The Parameter dialog box appears. Select from the job parameters that are defined for this job.
   - **Lookup Columns.** The Lookup Columns dialog box appears. Select a column from the input columns to the stage that you are using the SQL builder in.
   - **Expression Editor.** The Expression Editor opens. Build an expression that derives the value.
5. If you want to refine the update you are performing, choose a predicate from the **Predicate** list in the filter panel. Then use the expression editor to specify the actual filter (the fields displayed depend on the predicate you choose). For example, use the Comparison predicate to specify that a column should match a particular value, or the Between predicate to specify that a column falls within a particular range. The filter appears as a WHERE clause in the finished statement.
6. Click the **Add** button in the filter panel. The filter that you specify appears in the filter expression panel and is added to the update statement that you are building.
7. Click on the Sql tab to view the finished query.

---

### Building DELETE statements

Build DELETE statements to delete rows from a database table.
Procedure

1. Click the Delete tab.
2. Drag the table from which you want to delete rows from the repository tree to
   the canvas. You must have previously placed the table definitions in the IBM
   InfoSphere DataStage repository. The easiest way to do this is to import the
   definitions directly from your relational database.
3. You must choose an expression which defines the rows to be deleted. Choose a
   predicate from the Predicate list in the filter panel. Then use the expression
   editor to specify the actual filter (the fields displayed depend on the predicate
   you choose). For example, use the Comparison predicate to specify that a
   column should match a particular value, or the Between predicate to specify
   that a column falls within a particular range. The filter appears as a WHERE
   clause in the finished statement.
4. Click the Add button in the filter panel. The filter that you specify appears in
   the filter expression panel and is added to the update statement that you are
   building.
5. Click on the Sql tab to view the finished query.

The SQL builder Interface

The components in the top half of the SQL builder are common to all the types of
statement you can build. The bottom half comprises a series of tabbed pages. What
pages are available depends on the type of query you are building.

Toolbar

The SQL builder toolbar contains the following tools.

- **Clear Query** removes the field entries for the current SQL query.
- **Cut** removes items and placed them on the Microsoft Windows clipboard so
  they can be pasted elsewhere.
- **Copy** copies items and place them on the Windows clipboard so they can be
  pasted elsewhere.
- **Paste** pastes items from the Windows clipboard to certain places in the SQL
  builder.
- **SQL properties** opens the Properties dialog box.
- **Quoting** toggles quotation marks in table and column names in the generated
  SQL statements.
- **Validation** toggles the validation feature. Validation automatically occurs when
  you click OK to exit the SQL builder.
- **View Data** is available when you invoke the SQL builder from stages that
  support the viewing of data. It causes the calling stage to run the SQL as
  currently built and return the results for you to view.
- **Refresh** refreshes the contents of all the panels on the SQL builder.
- **Window View** allows you to select which panels are shown in the SQL builder
  window.
- **Help** opens the online help.

Tree Panel

This displays the table definitions that currently exist within the IBM InfoSphere
DataStage repository. The easiest way to get a table definition into the repository is
to import it directly from the database you want to query. You can do this via the
Designer client, or you can do it directly from the shortcut menu in the tree panel.
You can also manually define a table definition from within the SQL builder by selecting **New Table...** from the tree panel shortcut menu.

To select a table to query, select it in the tree panel and drag it to the table selection canvas. A window appears in the canvas representing the table and listing all its individual columns.

A shortcut menu allows you to:
- Refresh the repository view
- Define a new table definition (the Table Definition dialog box opens)
- Import metadata directly from a data source (a sub menu offers a list of source types)
- Copy a table definition (you can paste it in the table selection canvas)
- View the properties of the table definition (the Table Definition dialog box opens)

You can also view the properties of a table definition by double-clicking on it in the repository tree.

**Table Selection Canvas**

Drag a table from the tree panel to the table selection canvas. If the desired table does not exist in the repository, you can import it from the database you are querying by choosing **Import Metadata** from the tree panel shortcut menu.

The table appears in a window on the canvas, with a list of the columns and their types. For insert, update, and delete statements you can only place one table on the canvas. For select queries you can place multiple tables on the canvas.

Wherever you try to place the table on the canvas, the first table you drag will always be placed in the top left hand corner. If you are building a select query, subsequent tables can be dragged before or after the initial, or on a new row underneath. Eligible areas are highlighted on the canvas as you drag the table, and you can only drop a table in one of the highlighted areas. When you place tables on the same row, the SQL builder will automatically join the tables (you can alter the join if it's not what you want).

When you place tables on a separate row, no join is added. An old-style Cartesian product of the table rows on the different rows is produced: FROM FirstTable, SecondTable.

For details about joining tables, see [Joining Tables](#).

Click the **Select All** button underneath the table title bar to select all the columns in the table. Alternatively you can double-click on or drag individual columns from the table to the grid in the **Select**, **Insert**, or **Update** page to use just those columns in your query.

With a table selected in the canvas, a shortcut menu allows you to:
- Add a related table (select queries only). A submenu shows you tables that have a foreign key relationship with the currently selected one. Select a table to insert it in the canvas, together with the join expression inferred by the foreign key relationship.
- Remove the selected table.
• Select all the columns in the table (so that you could, for example, drag them all to the column selection grid).
• Open a Select Table dialog box to allow you to bind an alternative table for the currently selected table (select queries only).
• Open the Table Properties dialog box for the currently selected table.

With a join selected in the canvas (select queries only), a shortcut menu allows you to:
• Open the Alternate Relation dialog box to specify that the join should be based on a different foreign key relationship.
• Open the Join Properties dialog box to modify the type of join and associated join expression.

From the canvas background, a shortcut menu allows you to:
• Refresh the view of the table selection canvas.
• Paste a table that you have copied from the tree panel.
• View data - this is available when you invoke the SQL builder from stages that support the viewing of data. It causes the calling stage to run the SQL as currently built and return the results for you to view.
• Open the Properties dialog box to view details of the SQL syntax that the SQL builder is currently building a query for.

**Selection Page**

The Selection page appears when you are using the SQL builder to define a Select statement. Use this page to specify details of your select query. It has the following components.

**Column Selection Grid**

This is where you specify which columns are to be included in your query. You can populate the grid in a number of ways:
• drag columns from the tables in the table selection canvas.
• choose columns from a drop-down list in the grid.
• double-click the column name in the table selection canvas.
• copy and paste from the table selection canvas.

The grid has the following fields:

**Column expression**

Identifies the column to be included in the query. You can specify:
• **Job parameter.** A dialog box appears offering you a choice of available job parameters. This allows you to specify the value to be used in the query at run time (the stage you are using the SQL builder from must allow job parameters for this to appear).
• **Expression.** An expression editor dialog box appears, allowing you to specify an expression that represents the value to be used in the query.
• **Data flow variable.** A dialog box appears offering you a choice of available data flow variables (the stage you are using the SQL builder from must support data flow variables for this to appear)
• **Lookup Column.** You can directly select a column from one of the tables in the table selection canvas.
Table
Identifies the table that the column belongs to. If you populate the column grid by
dragging, copying or double-clicking on a column from the table selection canvas,
the table name is filled in automatically. You can also choose a table from the
drop-down list.

To specify the table name at runtime, choose a job parameter from the drop-down
list.

Column Alias
This allows you to specify an alias for the column.

Output
This is selected to indicate that the column will be output by the query. This is
automatically selected when you add a column to the grid.

Sort
Choose Ascending or Descending to have the query sort the returned rows by the
value of this column. Selecting to sort results in an ORDER BY clause being added
to the query.

Sort Order
 Allows you to specify the order in which rows are sorted if you are ordering by
more than one column.

Context Menu
A shortcut menu allows you to:
• Paste a column that you’ve copied from the table selection canvas.
• Insert a row in the grid.
• Show or hide the filter panel.
• Remove a row from the grid.

Filter Panel
The filter panel allows you to specify a WHERE clause for the SELECT statement
you are building. It comprises a predicate list and an expression editor panel, the
contents of which depends on the chosen predicate.

See Expression Editor for details on using the expression editor that the filter panel
provides.

Filter Expression Panel
This panel, at the bottom of the SQL builder window, displays any filters that you
have added to the query being built. You can edit the filter manually in this panel.
Alternatively you can type a filter straight in, without using the filter expression
editor.

Group Page
The Group page appears when you are using the SQL builder to define a select
statement. Use the Group page to specify that the results of a select query are
grouped by a column, or columns. Also, use it to aggregate the results in some of
the columns, for example, you could specify COUNT to count the number of rows
that contain a not-null value in a column.
The **Group** tab gives access to the toolbar, tree panel, and the table selection canvas, in exactly the same way as the Selection page.

**Grouping Grid**

This is where you specify which columns are to be grouped by or aggregated on.

The grid is populated with the columns that you selected on the Selection page. You can change the selected columns or select new ones, which will be reflected in the selection your query makes.

The grid has the following fields:

- **Column expression.** Identifies the column to be included in the query. You can modify the selections from the Selection page, or build a column expression.
  - Job parameter. A dialog box appears offering you a choice of available job parameters. This allows you to specify the value to be used in the query at run time (the stage you are using the SQL builder from must allow job parameters for this to appear).
  - Expression Editor. An expression editor dialog box appears, allowing you to specify an expression that represents the value to be used in the query.
  - Data flow variable. A dialog box appears offering you a choice of available data flow variables (the stage you are using the SQL builder from must support data flow variables for this to appear).
  - Lookup Column. You can directly select a column from one of the tables in the table selection canvas.

- **Column Alias.** This allows you to specify an alias for the column. If you select an aggregation operation for a column, SQL builder will automatically insert an alias of the form Alison; you can edit this if required.

- **Output.** This is selected to indicate that the column will be output by the query. This is automatically selected when you add a column to the grid.

- **Distinct.** Select this check box if you want to add the DISTINCT qualifier to an aggregation. For example, a COUNT aggregation with the distinct qualifier will count the number of rows with distinct values in a field (as opposed to just the not-null values). For more information about the DISTINCT qualifier, see **SQL Properties Dialog Box**.

- **Aggregation.** Allows you to select an aggregation function to apply to the column (note that this is mutually exclusive with the Group By option). See **Aggregation Functions** for details about the available functions.

- **Group By.** Select the check box to specify that query results should be grouped by the results in this column.

**Aggregation Functions**

The aggregation functions available vary according to the stage you have opened the SQL builder from. The following are the basic ones supported by all SQL syntax variants.

The following aggregation functions are supported.

- **AVG.** Returns the mean average of the values in a column. For example, if you had six rows with a column containing a price, the six rows would be added together and divided by six to yield the mean average. If you specify the DISTINCT qualifier, only distinct values will be averaged; if the six rows only contained four distinct prices then these four would be added together and divided by four to produce a mean average.
Filter Panel
The filter panel allows you to specify a HAVING clause for the SELECT statement you are building. It comprises a predicate list and an expression editor panel, the contents of which depends on the chosen predicate.

See [Expression Editor](#) for details on using the expression editor that the filter panel provides.

Filter Expression Panel
This panel displays any filters that you have added to the query being built. You can edit the filter manually in this panel. Alternatively you can type a filter straight in, without using the filter panel.

Insert Page
The Insert page appears when you are using the SQL builder to define an insert statement. Use this page to specify details of your insert statement. This page has the component **insert columns grid**.

Insert Columns Grid
This is where you specify which columns are to be included in your statement and what values they will take. The grid has the following fields:

**Insert Column**
Identifies the columns to be included in the statement. You can populate this in a number of ways:
- drag columns from the table in the table selection canvas.
- choose columns from a drop-down list in the grid.
- double-click the column name in the table selection canvas.
- copy and paste from the table selection canvas.

**Insert Value**
Identifies the values that you are setting the corresponding column to. You can specify one of the following in giving a value. You can also type a value directly into this field.
- **Job parameter**. A dialog box appears offering you a choice of available job parameters. This allows you to specify the value to be used in the query at run time (the stage you are using the SQL builder from must allow job parameters for this to appear).
- **Expression**. An expression editor dialog box appears, allowing you to specify an expression that represents the value to be used in the query.
• **Data flow variable.** A dialog box appears offering you a choice of available data flow variables (the stage you are using the SQL builder from must support data flow variables for this to appear)

• **Lookup Column.** You can directly select a column from one of the tables in the table selection canvas.

---

**Update Page**

The Update page appears when you are using the SQL builder to define an update statement. Use this page to specify details of your update statement. It has the following components.

**Update Column Grid**

This is where you specify which columns are to be included in your statement and what values they will take. The grid has the following fields:

**Update Column**
Identifies the columns to be included in the statement. You can populate this in a number of ways:
- drag columns from the table in the table selection canvas.
- choose columns from a drop-down list in the grid.
- double-click the column name in the table selection canvas.
- copy and paste from the table selection canvas.

**Update Value**
Identifies the values that you are setting the corresponding column to. You can specify one of the following in giving a value. You can also type a value directly into this field.
- **Job parameter.** A dialog box appears offering you a choice of available job parameters. This allows you to specify the value to be used in the query at run time (the stage you are using the SQL builder from must allow job parameters for this to appear).
- **Expression.** An expression editor dialog box appears, allowing you to specify an expression that represents the value to be used in the query.
- **Data flow variable.** A dialog box appears offering you a choice of available data flow variables (the stage you are using the SQL builder from must support data flow variables for this to appear)
- **Lookup Column.** You can directly select a column from one of the tables in the table selection canvas.

**Filter Panel**

The filter panel allows you to specify a WHERE clause for the update statement you are building. It comprises a predicate list and an expression editor panel, the contents of which depends on the chosen predicate.

See [Expression Editor](#) for details on using the expression editor that the filter panel provides.

**Filter Expression Panel**

This panel displays any filters that you have added to the query being built. You can edit the filter manually in this panel. Alternatively you can type a filter straight in, without using the filter panel.
Delete Page

The Delete page appears when you are using the SQL builder to define a delete statement. Use this page to specify details of your delete statement. It has the following components.

Filter Panel

The filter panel allows you to specify a WHERE clause for the delete statement you are building. It comprises a predicate list and an expression editor panel, the contents of which depends on the chosen predicate.

See "Expression Editor" for details on using the expression editor that the filter panel provides.

Filter Expression Panel

This panel displays any filters that you have added to the query being built. You can edit the filter manually in this panel. Alternatively you can type a filter straight in, without using the filter panel.

Sql Page

Click the Sql tab to view the generated statement. Using the shortcut menu, you can copy the statement for use in other environments.

For select queries, if the columns you have defined as output columns for your stage do not match the columns that the SQL statement is generating, use the Resolve columns grid to reconcile them. In most cases, the columns match.

Resolve Columns Grid

If the columns you have loaded onto your stage editor (the loaded columns) do not match the columns generated by the SQL statement (the result columns) you have defined, the Resolve columns grid gives you the opportunity to reconcile them. Ideally the columns should match (and in normal circumstances usually would). A mismatch would cause the metadata in your job to become out of step with the metadata as loaded from your source database (which could cause a problem if you are performing usage analysis based on that table).

If there is a mismatch, the grid displays a warning message. Click the Auto Match button to resolve the mismatch. You are offered the choice of matching by name, by order, or by both. When matching, the SQL builder seeks to alter the columns generated by the SQL statement to match the columns loaded onto the stage.

If you choose Name matching, and a column of the same name with a compatible data type is found, the SQL builder:

- Moves the result column to the equivalent position in the grid to the loaded column (this will change the position of the named column in the SQL).
- Modifies all the attributes of the result column to match those of the loaded column.

If you choose Order matching, the builder works through comparing each results column to the loaded column in the equivalent position. If a mismatch is found, and the data type of the two columns is compatible, the SQL builder:

- Changes the alias name of the result column to match the loaded column (provided the results set does not already include a column of that name).
• Modifies all the attributes of the result column to match those of the loaded column.

If you choose Both, the SQL builder applies Name matching and then Order matching.

If auto matching fails to reconcile the columns as described above, any mismatched results column that represents a single column in a table is overwritten with the details of the loaded column in the equivalent position.

When you click OK in the Sql tab, the SQL builder checks to see if the results columns match the loaded columns. If they don’t, a warning message is displayed allowing you to proceed or cancel. Proceeding causes the loaded columns to be merged with the results columns:
• Any matched columns are not affected.
• Any extra columns in the results columns are added to the loaded columns.
• Any columns in the loaded set that do not appear in the results set are removed.
• For columns that don’t match, if data types are compatible the loaded column is overwritten with the results column. If data types are not compatible, the existing loaded column is removed and replaced with the results column.

You can also edit the columns in the Results part of the grid in order to reconcile mismatches manually.

Expression Editor

The Expression Editor allows you to specify details of a WHERE clause that will be inserted in your select query or update or delete statement. You can also use it to specify WHERE clause for a join condition where you are joining multiple tables, or for a HAVING clause. A variant of the expression editor allows you to specify a calculation, function, or a case statement within an expression. The Expression Editor can be opened from various places in the SQL builder.

Main Expression Editor

To specify an expression:
• Choose the type of filter by choosing a predicate from the list.
• Fill in the information required by the Expression Editor fields that appear.
• Click the Add button to add the filter to the query you are building. This clears the expression editor so that you can add another filter if required.

The contents of the expression editor vary according to which predicate you have selected. The following predicates are available:
• **Between.** Allows you to specify that the value in a column should lay within a certain range.
• **Comparison.** Allows you to specify that the value in a column should be equal to, or greater than or less than, a certain value.
• **In.** Allows you to specify that the value in a column should match one of a list of values.
• **Like.** Allows you to specify that the value in a column should contain, start with, end with, or match a certain value.
• **Null.** Allows you to specify that a column should be null or should not be null.
Between
The expression editor when you have selected the Between predicate contains:

- **Column.** Choose the column on which you are filtering from the drop-down list. You can also specify:
  - **Job parameter.** A dialog box appears offering you a choice of available job parameters. This allows you to specify the value to be used in the query at run time (the stage you are using the SQL builder from must allow job parameters for this to appear).
  - **Expression.** An expression editor dialog box appears, allowing you to specify an expression that represents the value to be used in the query.
  - **Data flow variable.** A dialog box appears offering you a choice of available data flow variables (the stage you are using the SQL builder from must support data flow variables for this to appear).
  - **Column.** You can directly select a column from one of the tables in the table selection canvas.

- **Between/Not Between.** Choose Between or Not Between from the drop-down list to specify whether the value you are testing should be inside or outside your specified range.

- **Start of range.** Use this field to specify the start of your range. Click the menu button to the right of the field and specify details about the argument you are using to specify the start of the range, then specify the value itself in the field.

- **End of range.** Use this field to specify the end of your range. Click the menu button to the right of the field and specify details about the argument you are using to specify the end of the range, then specify the value itself in the field.

Comparison
The expression editor when you have selected the Comparison predicate contains:

- **Column.** Choose the column on which you are filtering from the drop-down list. You can specify one of the following in identifying a column:
  - **Job parameter.** A dialog box appears offering you a choice of available job parameters. This allows you to specify the value to be used in the query at run time (the stage you are using the SQL builder from must allow job parameters for this to appear).
  - **Expression.** An expression editor dialog box appears, allowing you to specify an expression that represents the value to be used in the query.
  - **Data flow variable.** A dialog box appears offering you a choice of available data flow variables (the stage you are using the SQL builder from must support data flow variables for this to appear).
  - **Column.** You can directly select a column from one of the tables in the table selection canvas.

- **Comparison operator.** Choose the comparison operator from the drop-down list. The available operators are:
  - `=` equals
  - `<>` not equal to
  - `<` less than
  - `<=` less than or equal to
  - `>` greater than
  - `>=` greater than or equal to

- **Comparison value.** Use this field to specify the value you are comparing to. Click the menu button to the right of the field and choose the data type for the value from the menu, then specify the value itself in the field.
In

The expression editor when you have selected the In predicate contains:

- **Column.** Choose the column on which you are filtering from the drop-down list. You can specify one of the following in identifying a column:
  - **Job parameter.** A dialog box appears offering you a choice of available job parameters. This allows you to specify the value to be used in the query at run time (the stage you are using the SQL builder from must allow job parameters for this to appear).
  - **Expression.** An expression editor dialog box appears, allowing you to specify an expression that represents the value to be used in the query.
  - **Data flow variable.** A dialog box appears offering you a choice of available data flow variables (the stage you are using the SQL builder from must support data flow variables for this to appear)
  - **Column.** You can directly select a column from one of the tables in the table selection canvas.
- **In/Not In.** Choose IN or NOT IN from the drop-down list to specify whether the value should be in the specified list or not in it.
- **Selection.** These fields allows you to specify the list used by the query. Use the menu button to the right of the single field to specify details about the argument you are using to specify a list item, then enter a value. Click the double right arrow to add the value to the list.
  To remove an item from the list, select it then click the double left arrow.

Like

The expression editor when you have selected the Like predicate is as follows. The fields it contains are:

- **Column.** Choose the column on which you are filtering from the drop-down list. You can specify one of the following in identifying a column:
  - **Job parameter.** A dialog box appears offering you a choice of available job parameters. This allows you to specify the value to be used in the query at run time (the stage you are using the SQL builder from must allow job parameters for this to appear).
  - **Expression.** An expression editor dialog box appears, allowing you to specify an expression that represents the value to be used in the query.
  - **Data flow variable.** A dialog box appears offering you a choice of available data flow variables (the stage you are using the SQL builder from must support data flow variables for this to appear)
  - **Column.** You can directly select a column from one of the tables in the table selection canvas.
- **Like/Not Like.** Choose LIKE or NOT LIKE from the drop-down list to specify whether you are including or excluding a value in your comparison.
- **Like Operator.** Choose the type of Like or Not Like comparison you want to perform from the drop-down list. Available operators are:
  - Match Exactly. Your query will ask for an exact match to the value you specify.
  - Starts With. Your query will match rows that start with the value you specify.
  - Ends With. Your query will match rows that end with the value you specify.
  - Contains. Your query will match rows that contain the value you specify anywhere within them.
- **Like Value.** Specify the value that your LIKE predicate will attempt to match.
Null
The expression editor when you have selected the Null predicate is as follows. The fields it contains are:

- **Column.** Choose the column on which you are filtering from the drop-down list. You can specify one of the following in identifying a column:
  - **Job parameter.** A dialog box appears offering you a choice of available job parameters. This allows you to specify the value to be used in the query at run time (the stage you are using the SQL builder from must allow job parameters for this to appear).
  - **Expression.** An expression editor dialog box appears, allowing you to specify an expression that represents the value to be used in the query.
  - **Data flow variable.** A dialog box appears offering you a choice of available data flow variables (the stage you are using the SQL builder from must support data flow variables for this to appear).
  - **Column.** You can directly select a column from one of the tables in the table selection canvas.
- **Is Null/Is Not Null.** Choose whether your query will match a NULL or NOT NULL condition in the column.

Join
This predicate is only available when you are building an Oracle 8i query with an 'old style' join expression. The Expression Editor is as follows.

- **Left column.** Choose the column to be on the left of your join from the drop-down list.
- **Join type.** Choose the type of join from the drop-down list.
- **Right column.** Choose the column to be on the right of your query from the drop-down list.

Calculation/Function/Case Expression Editor
This version of the expression editor allows you to specify an expression within a WHERE or HAVING expression, or a join condition. Expression Editor dialogs are numbered to show how deeply you are nesting them. Fields in the Expression Editor panel vary according to the chosen predicate as follows:

**Calculation**
The expression editor when you have selected the Calculation predicate contains these fields:

- **Left Value.** Enter the argument you want on the left of your calculation. You can choose the type of argument by clicking the menu button on the right and choosing a type from the menu.
- **Calculation Operator.** Choose the operator for your calculation from the drop-down list.
- **Right Value.** Enter the argument you want on the right of your calculation. You can choose the type of argument by clicking the menu button on the right and choosing a type from the menu.

**Functions**
The expression editor when you have selected the Functions predicate contains these fields:

- **Function.** Choose a function from the drop-down list. The list of available functions depends on the database you are building the query for.
• **Description.** Gives a description of the function you have selected.

• **Parameters.** Enter the parameters required by the function you have selected.
  The parameters that are required vary according to the selected function.

**Case**

The case option on the expression editor enables you to include case statements in the SQL you are building. You can build case statements with the following syntax.

```sql
CASE WHEN condition THEN value
CASE WHEN...
ELSE value
```

or

```sql
CASE subject
  WHEN match_value THEN value
  WHEN...
  ELSE value
```

The expression editor when you have selected the Case predicate contains these fields:

- **Case Expression.** This is the subject of the case statement. Specify this if you are using the second syntax described above (CASE subject WHEN). By default, the field offers a choice of the columns from the table or tables you have dragged to the table selection canvas. To choose an alternative, click the browse button next to the field. This gives you a choice of data types, or of specifying another expression, a function, or a job parameter.

- **When.** This allows you to specify a condition or match value for your case statement. By default, the field offers a choice of the columns from the table or tables you have dragged to the table selection canvas. To choose an alternative, click the browse button next to the field. This gives you a choice of data types, or of specifying another expression, a function, or a job parameter. You can access the main expression editor by choose case expression editor from the menu. This allows you to specify expressions such as comparisons. You would typically use this in the first syntax example. For example, you would specify grade=3 as the condition in the expression WHEN grade=3 THEN 'first class'.

- **Then.** Use this to specify the value part of the case expression. By default, the field offers a choice of the columns from the table or tables you have dragged to the table selection canvas. To choose an alternative, click the browse button next to the field. This gives you a choice of data types, or of specifying another expression, a function, or a job parameter.

- **Add.** Click this to add a case expression to the query. This clears the When and Then fields so that you can specify another case expression.

- **Else Expression.** Use this to specify the value for the optional ELSE part of the case expression.

**Expression Editor Menus**

A button appears to the right of many of the fields in the expression editor and related dialogs. Where it appears you can click it to open a menu that allows you to specify more details about an argument being given in an expression.

- **Bit.** Specifies that the argument is of type bit. The argument field offers a choice of 0 or 1 in a drop-down list.

- **Column.** Specifies that the argument is a column name. The argument field offer a choice of available columns in a drop-down list.
• **Date.** Specifies that the argument is a date. The SQL builder enters today’s date in the format expected by the database you are building the query for. You can edit this date as required or click the drop-down button and select from a calendar.

• **DateTime.** Specifies that the argument is a date time. The SQL builder inserts the current date and time in the format that the database the query is being built for expects. You can edit the date time as required.

• **Plaintext.** Allows you to select the default value of an argument (if one is defined).

• **Expression Editor.** You can specify a function or calculation expression as an argument of an expression. Selecting this causes the Calculation/Function version of the expression editor to open.

• **Function.** You can specify a function as an argument to an expression. Selecting this causes the Functions Form dialog box to open. The functions available depend on the database that the query you are building is intended for.

• **Job Parameter.** You can specify that the argument is a job parameter, the value for which is supplied when you actually run the IBM InfoSphere DataStage job. Selecting this opens the Parameters dialog box.

• **Integer.** Choose this to specify that the argument is of integer type.

• **String.** Select this to specify that the argument is of string type.

• **Time.** Specifies that the argument is the current local time. You can edit the value.

• **Timestamp.** Specifies that the argument is a timestamp. You can edit the value. The SQL builder inserts the current date and time in the format that the database that the query is being built for expects.

### Functions Form Dialog Box

This dialog box allows you to select a function for use within an expression, and specify parameters for the function.

The fields are as follows:

• **Function.** Choose a function from the drop-down list.
  
  The available functions depend on the database that you are building the query for.

• **Format.** Gives the format of the selected function as a guide.

• **Description.** Gives a description of the function you have selected.

• **Result.** Shows the actual function that will be included in the query as specified in this dialog box.

• **Parameters.** Enter the parameters required by the function you have selected.
  
  The parameters that are required vary according to the selected function.

### Function Dialog Box:

This dialog box allows you to select a function for use within an expression, and specify parameters for the function.

The fields are as follows:

• **Function.** Choose a function from the drop-down list.
  
  The available functions depend on the database that you are building the query for.
• **Format.** Gives the format of the selected function as a guide.

• **Description.** Gives a description of the function you have selected.

• **Result.** Shows the actual function that will be included in the query as specified in this dialog box.

• **Parameters.** Enter the parameters required by the function you have selected. The parameters that are required vary according to the selected function.

**Parameters Dialog Box**

This dialog box lists the job parameters that are currently defined for the job within which you are working. It also gives the data type of the parameter. Note that the SQL builder does not check that the type of parameter you are inserting matches the type expected by the argument you are using it for.

---

**Joining Tables**

When you use the SQL builder to help you build select queries, you can specify table joins within the query.

When you drag multiple tables onto the table selection canvas, the SQL builder attempts to create a join between the table added and the one already on the canvas to its left. If foreign key metadata is available for the tables, the SQL builder uses it. The join is represented by a line joining the columns the SQL builder has decided to join on. After the SQL builder automatically inserts a join, you can amend it.

When you add a table to the canvas, SQL builder determines how to join the table with tables that are on the canvas. The process depends on whether the added table is positioned to the right or left of the tables on the canvas.

To construct a join between the added table and the tables to its left:
1. SQL builder starts with the added table.
2. Determine if there is a foreign key between the added table and the subject table.
   - If a foreign key is present, continue to Step 3.
   - If a foreign key is not present, skip to Step 4.
3. Choose between alternatives for joining the tables that is based on the following precedence.
   - Relations that apply to the key fields of the added tables
   - Any other foreign key relation
   Construct an INNER JOIN between the two tables with the chosen relationship dictating the join criteria.
4. Take the subject as the next table to the left, and try again from step 2 until either a suitable join condition has been found or all tables, to the left, have been exhausted.
5. If no join condition is found among the tables, construct a default join.
   If the SQL grammar does not support a CROSS JOIN, an INNER JOIN is used with no join condition. Because this produces an invalid statement, you must set a suitable condition, either through the Join Properties dialog box, or by dragging columns between tables.
   An INNER JOIN is used with no join condition. Because this produces an invalid statement, you must set a suitable condition, either through the Join Properties dialog box, or by dragging columns between tables.
To construct a join between the added table and tables to its right:

1. SQL builder starts with the added table.
2. Determine if foreign key information exists between the added table and the subject table.
   - If a foreign key is present, continue to Step 3.
   - If a foreign key is not present, skip to Step 4.
3. Choose between alternatives based on the following precedence:
   - Relations that apply to the key fields of the added tables
   - Any other joins
   Construct an INNER JOIN between the two tables with the chosen relationship dictating the join criteria.
4. Take the subject as the next table to the right and try again from step 2.
5. If no join condition is found among the tables, construct a default join.
   If the SQL grammar does not support a CROSS JOIN, an INNER JOIN is used with no join condition. Because this produces an invalid statement, you must set a suitable condition, either through the Join Properties dialog box, or by dragging columns between tables.
   An INNER JOIN is used with no join condition. Because this produces an invalid statement, you must set a suitable condition, either through the Join Properties dialog box, or by dragging columns between tables.

### Specifying Joins

There are three ways of altering the automatic join that the SQL builder inserts when you add more than one table to the table selection canvas:

- Using the Join Properties dialog box. Open this by selecting the link in the table selection canvas, right clicking and choosing Properties from the shortcut menu. This dialog allows you to choose a different type of join, choose alternative conditions for the join, or choose a natural join.

- Using the Alternate Relation dialog box. Open this by selecting the link in the table selection canvas, right clicking and choosing Alternate Relation from the shortcut menu. This dialog allows you to change foreign key relationships that have been specified for the joined tables.

- By dragging a column from one table to another column in any table to its right on the canvas. This replaces the existing automatic join and specifies an equijoin between the source and target column. If the join being replaced is currently specified as an inner or outer join, then the type is preserved, otherwise the new join will be an inner join.

Yet another approach is specify the join using a WHERE clause rather than an explicit join operation (although this is not recommended where your database supports explicit join statements). In this case you would:

1. Specify the join as a Cartesian product. (SQL builder does this automatically if it cannot determine the type of join required).
2. Specify a filter in the Selection tab filter panel. This specifies a WHERE clause that selects rows from within the Cartesian product.

If you are using the SQL builder to build Oracle 8i, Microsoft SQL Server, IBM Informix®, or Sybase queries, you can use the Expression Editor to specify a join condition, which will be implemented as a WHERE statement. Oracle 8i does not support JOIN statements.
**Join Properties Dialog Box**

This dialog box allows you to change the type of an existing join and modify or specify the join condition.

The dialog box contains the following fields:

- **Cartesian product.** The Cartesian product is the result that is returned from two or more tables that are selected from, but not joined; that is, no join condition is specified. The output is all possible rows from all the tables selected from. For example, if you selected from two tables, the database would pair every row in the first table with every row in the second table. If each table had 6 rows, the Cartesian product would return 36 rows.

  If the SQL builder cannot insert an explicit join based on available information, it will default to a Cartesian product that is formed with the CROSS JOIN syntax in the FROM clause of the resulting SQL statement: `FROM FirstTable CROSS JOIN SecondTable`. You can also specify a Cartesian product by selecting the Cartesian product option in the Join Properties dialog box. The cross join icon is shown on the join.

- **Table join.** Select the **Table Join** option to specify that your query will contain join condition for the two tables being joined. The **Join Condition** panel is enabled, allowing you to specify further details about the join.

- **Join Condition panel.** This shows the expression that the join condition will contain. You can enter or edit the expression manually or you can use the menu button to the right of the panel to specify a natural join, open the Expression Editor, or open the Alternate relation dialog box.

- **Include.** These fields allow you to specify that the join should be an outer join, where the result of the query should include the rows as specified by one of the following:
  - Select **All rows from left table name** to specify a left outer join
  - Select **All rows from right table name** to specify a right outer join
  - Select both **All rows from left table name** and **All rows from right table name** to specify a full outer join

- **Join Icon.** This tells you the type of join you have specified.

**Alternate Relation Dialog Box**

This dialog box displays all the foreign key relationships that have been defined between the target table and other tables that appear to the left of it in the table selection canvas. You can select the relationship that you want to appear as the join in your query by selecting it so that it appears in the list box, and clicking **OK**.

---

**Properties Dialogs**

Depending where you are in the SQL builder, choosing **Properties** from the shortcut menu opens a dialog box as follows:

- **The Table Properties dialog box** opens when you select a table in the table selection canvas and choose **Properties** from the shortcut menu.

- **The SQL Properties dialog box** opens when you select the **Properties** icon in the toolbox or **Properties** from the table selection canvas background.

- **The Join Properties dialog box** opens when you select a join in the table selection canvas and choose **Properties** from the shortcut menu. This dialog is described in **Join Properties Dialog Box**.
Table Properties Dialog Box

The Table Properties dialog box contains the following fields:

- **Table name.** The name of the table whose properties you are viewing.
  You can click the menu button and choose Job Parameter to open the Parameter dialog box (see Parameters Dialog Box). This allows you to specify a job parameter to replace the table name if required, but note that the SQL builder will always refer to this table using its alias.

- **Alias.** The alias that the SQL builder uses to refer to this table. You can edit the alias if required. If the table alias is used in the selection grid or filters, changing the alias in this dialog box will update the alias there.

SQL Properties Dialog Box

This dialog box gives you details about the SQL grammar that the SQL builder uses. It contains the following fields:

- **Description.** The name and version of the SQL grammar.
  The SQL grammar depends on the stage that you invoke the SQL builder from.

- **DISTINCT.** Specify whether the SQL builder supports the DISTINCT qualifier.
  If the stage supports it, the DISTINCT option is selected.
Appendix A. Product accessibility

You can get information about the accessibility status of IBM products.

The IBM InfoSphere Information Server product modules and user interfaces are not fully accessible. The installation program installs the following product modules and components:

- IBM InfoSphere Business Glossary
- IBM InfoSphere Business Glossary Anywhere
- IBM InfoSphere DataStage
- IBM InfoSphere FastTrack
- IBM InfoSphere Information Analyzer
- IBM InfoSphere Information Services Director
- IBM InfoSphere Metadata Workbench
- IBM InfoSphere QualityStage

For information about the accessibility status of IBM products, see the IBM product accessibility information at [http://www.ibm.com/able/product_accessibility/index.html](http://www.ibm.com/able/product_accessibility/index.html).

Accessible documentation

Accessible documentation for InfoSphere Information Server products is provided in an information center. The information center presents the documentation in XHTML 1.0 format, which is viewable in most Web browsers. XHTML allows you to set display preferences in your browser. It also allows you to use screen readers and other assistive technologies to access the documentation.

The documentation that is in the information center is also provided in PDF files, which are not fully accessible.

IBM and accessibility

See the [IBM Human Ability and Accessibility Center](http://www.ibm.com/able/) for more information about the commitment that IBM has to accessibility.
Appendix B. Reading command-line syntax

This documentation uses special characters to define the command-line syntax.

The following special characters define the command-line syntax:

\[
\]
Identifies an optional argument. Arguments that are not enclosed in brackets are required.

\[
...
\]
Indicates that you can specify multiple values for the previous argument.

\[
|
\]
Indicates mutually exclusive information. You can use the argument to the left of the separator or the argument to the right of the separator. You cannot use both arguments in a single use of the command.

\[
\}
\]
Delimits a set of mutually exclusive arguments when one of the arguments is required. If the arguments are optional, they are enclosed in brackets ([ ]).

Note:
- The maximum number of characters in an argument is 256.
- Enclose argument values that have embedded spaces with either single or double quotation marks.

For example:

`wsetsrc [-S server] [-l label] [-n name] source`

The `source` argument is the only required argument for the `wsetsrc` command. The brackets around the other arguments indicate that these arguments are optional.

`wlsac [-l | -f format] [key...] profile`

In this example, the `-l` and `-f format` arguments are mutually exclusive and optional. The `profile` argument is required. The `key` argument is optional. The ellipsis (...) that follows the `key` argument indicates that you can specify multiple key names.

`wrb -import {rule_pack | rule_set}...`

In this example, the `rule_pack` and `rule_set` arguments are mutually exclusive, but one of the arguments must be specified. Also, the ellipsis marks (...) indicate that you can specify multiple rule packs or rule sets.
Appendix C. How to read syntax diagrams

The following rules apply to the syntax diagrams that are used in this information:

- Read the syntax diagrams from left to right, from top to bottom, following the path of the line. The following conventions are used:
  - The >>> symbol indicates the beginning of a syntax diagram.
  - The ---> symbol indicates that the syntax diagram is continued on the next line.
  - The >--- symbol indicates that a syntax diagram is continued from the previous line.
  - The --->< symbol indicates the end of a syntax diagram.
- Required items appear on the horizontal line (the main path).

```
  >>>---required_item---
```

- Optional items appear below the main path.

```
  >>>---required_item---optional_item---
```

If an optional item appears above the main path, that item has no effect on the execution of the syntax element and is used only for readability.

```
  >>>---required_item---optional_item---
```

- If you can choose from two or more items, they appear vertically, in a stack. If you must choose one of the items, one item of the stack appears on the main path.

```
  >>>---required_item---required_choice1---required_choice2---
```

If choosing one of the items is optional, the entire stack appears below the main path.

```
  >>>---required_item---optional_choice1---optional_choice2---
```

If one of the items is the default, it appears above the main path, and the remaining choices are shown below.

```
  >>>---required_item---default_choice---
```

- An arrow returning to the left, above the main line, indicates an item that can be repeated.
If the repeat arrow contains a comma, you must separate repeated items with a comma.

A repeat arrow above a stack indicates that you can repeat the items in the stack.

- Sometimes a diagram must be split into fragments. The syntax fragment is shown separately from the main syntax diagram, but the contents of the fragment should be read as if they are on the main path of the diagram.

**Fragment-name:**

- Keywords, and their minimum abbreviations if applicable, appear in uppercase. They must be spelled exactly as shown.
- Variables appear in all lowercase italic letters (for example, \textit{column-name}). They represent user-supplied names or values.
- Separate keywords and parameters by at least one space if no intervening punctuation is shown in the diagram.
- Enter punctuation marks, parentheses, arithmetic operators, and other symbols, exactly as shown in the diagram.
- Footnotes are shown by a number in parentheses, for example (1).
Appendix D. Contacting IBM

You can contact IBM for customer support, software services, product information, and general information. You also can provide feedback to IBM about products and documentation.

The following table lists resources for customer support, software services, training, and product and solutions information.

Table 19. IBM resources

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description and location</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM Support Portal</td>
<td>You can customize support information by choosing the products and the topics that interest you at <a href="http://www.ibm.com/support/entry/portal/Software/Information_Management/InfoSphere_Information_Server">www.ibm.com/support/entry/portal/Software/Information_Management/InfoSphere_Information_Server</a></td>
</tr>
<tr>
<td>Software services</td>
<td>You can find information about software, IT, and business consulting services, on the solutions site at <a href="http://www.ibm.com/businesssolutions/">www.ibm.com/businesssolutions/</a></td>
</tr>
<tr>
<td>My IBM</td>
<td>You can manage links to IBM Web sites and information that meet your specific technical support needs by creating an account on the My IBM site at <a href="http://www.ibm.com/account/">www.ibm.com/account/</a></td>
</tr>
<tr>
<td>Training and certification</td>
<td>You can learn about technical training and education services designed for individuals, companies, and public organizations to acquire, maintain, and optimize their IT skills at <a href="http://www.ibm.com/software/sw-training/">http://www.ibm.com/software/sw-training/</a></td>
</tr>
</tbody>
</table>
Appendix E. Accessing and providing feedback on the product documentation

Documentation is provided in a variety of locations and formats, including in help that is opened directly from the product client interfaces, in a suite-wide information center, and in PDF file books.

The information center is installed as a common service with IBM InfoSphere Information Server. The information center contains help for most of the product interfaces, as well as complete documentation for all the product modules in the suite. You can open the information center from the installed product or from a Web browser.

Accessing the information center

You can use the following methods to open the installed information center.

- Click the Help link in the upper right of the client interface.

  Note: From IBM InfoSphere FastTrack and IBM InfoSphere Information Server Manager, the main Help item opens a local help system. Choose Help > Open Info Center to open the full suite information center.

- Press the F1 key. The F1 key typically opens the topic that describes the current context of the client interface.

  Note: The F1 key does not work in Web clients.

- Use a Web browser to access the installed information center even when you are not logged in to the product. Enter the following address in a Web browser: http://host_name:port_number/infocenter/topic/com.ibm.swg.im.iis.productization.iisinfsv.home.doc/ic-homepage.html. The host_name is the name of the services tier computer where the information center is installed, and port_number is the port number for InfoSphere Information Server. The default port number is 9080. For example, on a Microsoft® Windows® Server computer named iisdocs2, the Web address is in the following format: http://iisdocs2:9080/infocenter/topic/com.ibm.swg.im.iis.productization.iisinfsv.nav.doc/dochome/iisinfsrv_home.html.

A subset of the information center is also available on the IBM Web site and periodically refreshed at http://pic.dhe.ibm.com/infocenter/iisinfsv/v9r1/index.jsp.

Obtaining PDF and hardcopy documentation

- A subset of the PDF file books are available through the InfoSphere Information Server software installer and the distribution media. The other PDF file books are available online and can be accessed from this support document: https://www.ibm.com/support/docview.wss?uid=swg27008803&wv=1

- You can also order IBM publications in hardcopy format online or through your local IBM representative. To order publications online, go to the IBM Publications Center at http://www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss
Providing comments on the documentation

Your feedback helps IBM to provide quality information. You can use any of the following methods to provide comments:

- To comment on the information center, click the Feedback link on the top right side of any topic in the information center.
- Send your comments by using the online readers' comment form at [www.ibm.com/software/awdtools/rcf/](http://www.ibm.com/software/awdtools/rcf/)
- Send your comments by e-mail to comments@us.ibm.com. Include the name of the product, the version number of the product, and the name and part number of the information (if applicable). If you are commenting on specific text, include the location of the text (for example, a title, a table number, or a page number).
Notices and trademarks

This information was developed for products and services offered in the U.S.A.

Notices

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user’s responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web
sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to
IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

**Trademarks**

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

The following terms are trademarks or registered trademarks of other companies:

Adobe is a registered trademark of Adobe Systems Incorporated in the United States, and/or other countries.

Intel and Itanium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows and Windows NT are trademarks of Microsoft Corporation in the United States, other countries, or both.

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

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

The United States Postal Service owns the following trademarks: CASS, CASS Certified, DPV, LACS[^1], ZIP, ZIP + 4, ZIP Code, Post Office, Postal Service, USPS and United States Postal Service. IBM Corporation is a non-exclusive DPV and LACS[^1] licensee of the United States Postal Service.

Other company, product or service names may be trademarks or service marks of others.
Index

A
Access method property 29
Account property 30
adding deprecated stages to palette 6
After SQL (node) property 31
After SQL file property 30
After SQL property 30
Allow duplicate rows property 32
apiconfig.xml file setting 25
Array size property 32
Before SQL (node) property 34
Before SQL file property 33
Before SQL property 33
Before/After SQL property 33
BTEQ export utility 107
Buffer usage property 35
Bulk access property 35
bulk loading tables 97, 107

B
Before SQL (node) property 34
Before SQL file property 33
Before SQL property 33
Before/After SQL property 33
BTEQ export utility 107
Buffer usage property 35
Bulk access property 35
bulk loading tables 97, 107

C
cannot find on palette 6
Case sensitive property 35
CC_TERA_DEBUG environment variable 127
CC_TERA_FAIL_ON_DBS_CRASH environment variable 127
CC_TERA_MAX_DECIMAL_RETURN environment variable 127
CC_TERA_OPTIMIZE_CONNECTIONS environment variable 127
CC_TERA_RECONNECT_FLAG environment variable 127
CC_TERA_TMSM_LOG_EVENT environment variable 127
CC_TERA_USE_ISO88591_ASCII_MAP environment variable 127
Character set property 35
case sets 28, 29
Checkpoint timeout property 36
Cleanup mode property 36
Client character set property 36
Client character set, options 28
Column delimiter 37
column values 37
Columns property 37
command-line syntax conventions 155
commands syntax 155
configuration Teradata connector 25
connection properties Account 30
Client character set 36
Database 38
connection properties (continued)
password 50
Server 53
user name 60
connector migration tool 1
Connector Migration Tool command line interface 4
Connector Migration Tool user interface 2
containers 1
migrating to use connectors 2, 4
conversion codes
NLSmapname 50
Create statement property 38
Create table options property 38
Create Table Properties dialog box Teradata API stages 69
customer support contacting 159

D
Data block size property 38
data types Teradata connector 21
Database property 38
deadlocks 17
Delete multiple rows property 38
Delete statement property 39
deprecated stages 6
design time services
generating SQL statements at design time 20
validating SQL statements at design time 21
Drop statement property 39
DS NO_PROCESS_METADATA environment variable 127
dsn env script 10
Dual Load 25
Duplicate insert rows property 40
Duplicate update rows property 40

E
Enable LOB references property 40
Enable quoted identifiers property 40
End of data property 41
End of wave property 41
End row property 32, 41
End timeout property 41
environment variables required 25
Teradata 127
Teradata API stages 64, 98, 112
Teradata Load stages 64, 98, 112
Teradata MultiLoad/TPump/FastExport stages 64, 98, 112
Error control property 41
Error limit property 41

F
Fail on error property 42
Fail on size mismatch property 42
FastExport script, building 120
FastExport utility 111, 112
FastLoad utility 97, 100
File property 43
Free space percent property 43

G
Generate create statement at runtime property 43
Generate drop statement at runtime property 43
Generate SQL property 43
Generate truncate statement at runtime property 44
Generate UOWID property 44

I
Immediate access property 44
Insert statement property 44
Isolation level property 45

J
jmsTransportProvider.properties file location of 25
jobs 1
migrating to use connectors 2, 4

K
Key column property 46

L
legal notices 163
library path environment variable TMSM_HOME environment variable setting 25
Limit settings property 46
Load type property 46
Log column values 47
key values 47
Log table property 47
Logging Column delimiter 46
column values 46
Logon mechanism property 47
Macro database property 47
Make duplicate copies property 48
mapping string data 65, 99, 113
Max buffer size property 48
Max partition sessions property 48
Max sessions property 48
Maximum bytes per character
migrating to use connectors 1
Min sessions property 49
Missing delete rows property 49
Missing update rows property 50
MultiLoad script, building 114
MultiLoad utility 111

N
NLSmapname conversion 50
not on palette 6

O
Options tab
 Teradata API stages
 input link 69

P
Pack size property 50
palette
displaying stages 6
parallel canvas 65, 98, 112
parallel engine
 connecting to Teradata databases 9
Parallel synchronization property 50
parallel synchronization table
Teradata connector 22
Password property 50
Primary index type property 51
product accessibility
 accessiblity 153
product documentation
 accessing 161
Progress interval property 51
properties
 Access method 29
 Account 30
 After SQL 30
 After SQL (node) 31
 After SQL file 30
 Allow duplicate rows 32
 Array size 32
 Before SQL 33
 Before SQL (node) 34
 Before SQL file 33
 Before/After SQL 33
 Buffer usage 35
 Bulk access 35
 Character set 35
 Checkpoint timeout 36
 Cleanup mode 36
 Client character set 36
 Columns 37
 Create statement 38
Create table options 38
Data block size 38
Database 38
Delete multiple rows 38
Delete statement 39
Drop statement 39
Duplicate insert rows 40
Duplicate update rows 40
Enable LOB references 40
Enable quoted identifiers 40
End of data 41
End of wave 41
End row 32, 41
End timeout 41
Error control 41
Error limit 41
Error table 1 42
Error table 2 42
Fail on error 42
Fail on size mismatch 42
File 43
Free space percent 43
Generate create statement at
 runtime 43
Generate drop statement at
 runtime 43
Generate SQL 43
Generate truncate statement at
 runtime 44
Generate UOWID 44
Immediate access 44
Insert statement 44
Isolation level 45
Limit settings 46
Load type 46
Log table 47
Logon mechanism 47
Macro database 47
Make duplicate copies 48
Max buffer size 48
Max partition sessions 48
Max sessions 48
Maximum bytes per character 49
Min sessions 49
Missing delete rows 49
Missing update rows 50
Pack size 50
Parallel synchronization 50
Password 50
Primary index type 51
Progress interval 51
Queryband expression 51
Record count 51
Request type 52
Robust 52
Schema Reconciliation 52
Select statement 52
Serialize 53
Server 53
Server character set 53
Session 53
Sleep 53
SQL 53
Start mode 53
Start row 54
Statements 54

Q
Queryband expression property 51

R
reading from an export file 107, 108
reading from tables 73, 78
Record count property 51
reject row handling 79
Request type property 52
Robust property 52

S
Scheme Reconciliation property 52
Select statement property 52
Serialize property 53
Server character set property 53
Server property 53
server stages
 SQL builder 78
Session property 53
setting environment variables for
databases
 setting 10
Sleep property 53
software services
 contacting 159
Sort order property 53
special characters
 in command-line syntax 155
SQL builder
  server stages 78
SQL Builder
  Teradata API stages 78
  SQL property 53
SQL statements
  Teradata connector 13
  stage not on palette 6
  stages
  adding to palette 6
  Start mode property 53
  Start row property 54
  Statements property 54
status messages 123
stored procedures 64
Stream load property 54
support
customer 159
Sync database property 54
Sync ID property 54
Sync password property 54
Sync poll property 55
Sync server property 55
Sync table action property 55
Sync table cleanup property 56
Sync table property 55
Sync table write mode property 57
Sync timeout property 56
Sync user property 56
syntax
  command-line 155

Table action property 57
Table free space property 57
Table name property 58
Tenacity property 58
Teradata API stages
  additional SQL SELECT clauses 77
  character set mapping 67
  Columns tab
    Input page 70
    Output page 75
Create Table Properties dialog box 69
Data Browser 68, 73
data types 79
environment variables 64, 98, 112
FROM clause 77
functionality 66
General tab
  Output page 74
  Stage page 67, 68
  generated queries 76
  generated SQL queries 75, 77
  generated SQL statements 72
  GROUP BY clause 77
HAVING clause 77
input link
  Options tab 69
  input links 64
  Input page 66, 68, 71
  General tab 68
  introduction 63
  loading modes 101
  mapping data types 79
NLS tab 66, 67
Teradata API stages (continued)
  Options tab
    Output page 74
  ORDER BY clause 77
  output links 64
  Output page 67, 73
  parallel canvas 65, 98, 112
  Query Type 68
  reference links 64
  reject row handling 79
  Selection tab 75
  SQL 78
  SQL builder 78
  SQL Builder 68
SQL queries
  generated 75, 77
SQL statements
  FROM clause 77
  GROUP BY clause 77
  HAVING clause 77
  ORDER BY clause 77
  WHERE clause 77
SQL statements, syntax 76
SQL tab
  Input page 70
  Stage page 66
  Unicode 79
  update action 68
  user-defined SQL statements 72
  using generated queries 76
  WHERE clause 77
Teradata Call-Level Interface (CLI) 13, 14
Teradata connector
  avoiding deadlocks 17
  configuration 14
    as a source 16
    as a target 17
  logon mechanism 18
  Logon Mechanism 18
  to request data 19
  data types 21
  functionality 13
  load types for append 17
  overview 13
  parallel synchronization table 22
  performance 13
  sample job
    Sample job with TMSM integration 26
  SQL statements 13
  TMSM 25
Teradata connector support
  TMSM 27
Teradata databases
  setting up connectivity 9
Teradata Enterprise stage (continued)
  Properties tab
    Input page 88, 91
    Output page 93, 95
  Stage page 87
  Teradata Tools and Utilities, installing 82
Teradata Load stage
  automatic loading 102
  BTEQ export utility 107
  character set mapping 100
  Columns tab
    Input page 107
    Output page 109
commands
  DEFINE 102
  INSERT 102
  SET RECORD 102
data types 109
  Files tab 104
  functionality 99
  General tab
    Output page 108
    Stage page 100
  input data 102
  input links 97
  Input page 100, 103, 107
  installation 97
  introduction 97
  Limit Setting tab 106
  Load tab 103
  NLS tab 100
  output links 97, 107
  Output page 100, 108, 109
  Script tab 106
  Stage page 100
Teradata Load stages
  environment variables 64, 98, 112
  parallel canvas 65, 98, 112
Teradata Multi-System Manager
  Integration 25
Teradata MultiLoad/TPump/FastExport stage
  After tab 118
  Before tab 118
  Columns tab
    Input page 119
    Output page 123
  compiling a job 119, 123
data types 125
  FastExport tab 122
  functionality 113
  General tab
    Input page 115
    Output page 121
  Stage page 114, 121
  Generated tab 118
  input links 111
  Input page 114
  installation 112
  introduction 111
  Limit Settings tab 117
  MultiLoad/TPump tab 116
  output links 112
  Output page 121
  properties, defining 121
  running a job 119, 123
Teradata MultiLoad/TPump/FastExport
stage (continued)
  Script tab  118
  Selection tab  123
  SQL tab  123
  TPump tab  117
Teradata MultiLoad/TPump/FastExport stages
  environment variables  64, 98, 112
  parallel canvas  65, 98, 112
Teradata Parallel Transporter  13, 14
  operators  14
Teradata Tools and Utilities  82
TMSM event property  58
TMSM_ENABLED environment variable
  setting  25
TPump script, building  114
TPump utility  111
tracing  123
trademarks
  list of  163
Transaction mode property  58
Transaction property  58
Truncate statement property  59

U
Unused field action property  59
UOW class property  59
UOW source property  60
UOWID property  60
Update load property  60
Update statement property  60
User name property  60
User-defined SQL property  60

W
web sites
  non-IBM  157
Work table property  61
Write mode property  61
writing to tables  68, 73