Informatica® PowerExchange for Teradata Parallel Transporter API

10.2 HotFix 1

User Guide

Informatica PowerExchange for Teradata Parallel Transporter API User Guide
10.2 HotFix 1
August 2018

Publication Date: 2019-04-10

Table of Contents

Preface

The *Informatica PowerExchange® for Teradata Parallel Transporter API User Guide* provides information about reading data from and loading data to Teradata tables. It is written for database administrators and developers who create mappings to read data from or load data to Teradata.

This book assumes that you have knowledge of Teradata databases, Teradata Parallel Transporter, Teradata Connector for Hadoop, Hortonworks Connector for Teradata, Cloudera Connector Powered by Teradata, Sqoop, Informatica Developer, Informatica Big Data Management™, and the database engines and systems in your environment.

Informatica Resources

Informatica Network

Informatica Network hosts Informatica Global Customer Support, the Informatica Knowledge Base, and other product resources. To access Informatica Network, visit https://network.informatica.com.

As a member, you can:

- Access all of your Informatica resources in one place.
- Search the Knowledge Base for product resources, including documentation, FAQs, and best practices.
- View product availability information.
- Review your support cases.
- Find your local Informatica User Group Network and collaborate with your peers.

Informatica Knowledge Base

Use the Informatica Knowledge Base to search Informatica Network for product resources such as documentation, how-to articles, best practices, and PAMs.

To access the Knowledge Base, visit https://kb.informatica.com. If you have questions, comments, or ideas about the Knowledge Base, contact the Informatica Knowledge Base team at KB_Feedback@informatica.com.

Informatica Documentation

To get the latest documentation for your product, browse the Informatica Knowledge Base at https://kb.informatica.com/_layouts/ProductDocumentation/Page/ProductDocumentSearch.aspx.

If you have questions, comments, or ideas about this documentation, contact the Informatica Documentation team through email at infa_documentation@informatica.com.

Informatica Product Availability Matrixes

Product Availability Matrixes (PAMs) indicate the versions of operating systems, databases, and other types of data sources and targets that a product release supports. If you are an Informatica Network member, you can access PAMs at https://network.informatica.com/community/informatica-network/product-availability-matrices.

Informatica Velocity

Informatica Velocity is a collection of tips and best practices developed by Informatica Professional Services. Developed from the real-world experience of hundreds of data management projects, Informatica Velocity represents the collective knowledge of our consultants who have worked with organizations from around the world to plan, develop, deploy, and maintain successful data management solutions.

If you are an Informatica Network member, you can access Informatica Velocity resources at http://velocity.informatica.com.

If you have questions, comments, or ideas about Informatica Velocity, contact Informatica Professional Services at ips@informatica.com.

Informatica Marketplace

The Informatica Marketplace is a forum where you can find solutions that augment, extend, or enhance your Informatica implementations. By leveraging any of the hundreds of solutions from Informatica developers and partners, you can improve your productivity and speed up time to implementation on your projects. You can access Informatica Marketplace at https://marketplace.informatica.com.

Informatica Global Customer Support

You can contact a Global Support Center by telephone or through Online Support on Informatica Network.

To find your local Informatica Global Customer Support telephone number, visit the Informatica website at the following link:
http://www.informatica.com/us/services-and-training/support-services/global-support-centers.

If you are an Informatica Network member, you can use Online Support at http://network.informatica.com.

CHAPTER 1

Introduction to PowerExchange for Teradata Parallel Transporter API

This chapter includes the following topic:

PowerExchange for Teradata Parallel Transporter API Overview

You can use PowerExchange for Teradata Parallel Transporter API to read data from or write data to a Teradata database. You can run Teradata mappings in the native or Hadoop environment. If you choose the Hadoop environment, you can run Teradata mappings on the Blaze or Hive engines. You can configure dynamic mappings with Teradata sources and targets and run the mappings in the native environment.

PowerExchange for Teradata Parallel Transporter API uses different integration methods based on the environment in which you run the mappings.

Integration with Teradata Parallel Transporter API

When you run Teradata mappings in the native environment or on the Hive engine to read or load data in bulk, PowerExchange for Teradata Parallel Transporter API integrates with Teradata Parallel Transporter API (Teradata PT API). The Data Integration Service uses the Teradata PT API infrastructure to connect to Teradata.

The Data Integration Service uses the Export system operator to read data. The Export operator exports large amounts of data from Teradata tables or views.

The Data Integration Service uses one of the following Teradata PT API system operators to load data in bulk:

Load

Loads data in bulk into an empty Teradata table.

Stream

Performs insert, update, upsert, and delete operations against Teradata tables in near real time.

Update

Performs insert, update, upsert, and delete operations against Teradata tables.

Integration with Sqoop and Hortonworks Connector for Teradata

When you run Teradata mappings on a Hortonworks cluster and on the Blaze engine, the Data Integration Service invokes the Hortonworks Connector for Teradata at run time. The Data Integration Service then runs the mapping through Sqoop.

Integration with Sqoop and Cloudera Connector Powered by Teradata

When you run Teradata mappings on a Cloudera cluster and on the Blaze engine, the Data Integration Service invokes the Cloudera Connector Powered by Teradata at run time. The Data Integration Service then runs the mapping through Sqoop.

Integration with Teradata Connector for Hadoop

When you run Teradata mappings on the Hive engine, you can enable Teradata Connector for Hadoop (TDCH) to run the mapping and increase performance. TDCH is a set of API and tools that Teradata Corporation provides for parallel processing of data between Teradata databases and the Hadoop ecosystem of products.

Example

In a native environment, you can store payroll information of an organization for the past five years. You can use PowerExchange for Teradata Parallel Transporter API in the native environment to read payroll information and then write the data to Teradata tables for storage.

In a Hadoop environment, you can gather, store, and analyze large volumes of unstructured data such as web logs. You can process these large amounts of data in a Hadoop environment and use PowerExchange for Teradata Parallel Transporter API to write meaningful data to Teradata tables for analysis.

CHAPTER 2

PowerExchange for Teradata Parallel Transporter API Configuration

This chapter includes the following topics:

PowerExchange for Teradata Parallel Transporter API Configuration Overview

PowerExchange for Teradata Parallel Transporter API installs with the Informatica services and clients.

To configure PowerExchange for Teradata Parallel Transporter API, perform the following tasks:

- Verify the prerequisites.
- Configure environment variables.
- Verify database privileges.
- Configure JDBC.
- If you run Teradata mappings on the Blaze engine and on a Hortonworks cluster, configure Sqoop and Hortonworks Connector for Teradata.
- If you run Teradata mappings on the Blaze engine and on a Cloudera cluster, configure Sqoop and Cloudera Connector Powered by Teradata.
- If you run Teradata mappings on the Hive engine, configure Teradata Connector for Hadoop (TDCH) to increase performance.

Prerequisites

Before you use PowerExchange for Teradata Parallel Transporter API, you must perform the following tasks:

1. Install the Informatica services.
2. Install the Informatica clients. When you install the Informatica clients, the Developer tool is installed.
3. Create a Data Integration Service and a Model Repository Service in the Informatica domain.
4. To run mappings in the native environment, perform the following steps:
   - Install Teradata Parallel Transporter on the node that runs the Data Integration Service. You must update the Teradata Parallel Transporter installation to the latest patch.
   - Install the Teradata JDBC driver. You can download the latest version of the Teradata JDBC driver from the following URL: http://downloads.teradata.com/download/connectivity/jdbc-driver
   - To import metadata from Teradata tables, copy the Teradata JDBC jars to the following directory on the node where you installed the Developer tool:
     `<Informatica installation directory>/clients/externaljdbcjars`
   - Update the CLASSPATH environment variable to include the Teradata JDBC jars.
   - To run partitioned mappings, copy the Teradata JDBC jars to the following directory on the node that runs the Data Integration Service:
     `<Informatica installation directory>/externaljdbcjars`
5. To run Teradata mappings on the Blaze engine, perform the following steps:
   - Install Informatica Big Data Management on every node of the compute cluster.
   - Specify the Data Integration Service user name in the **Impersonation User Name** field of the Hadoop connection.
6. To run dynamic mappings in the native environment, you must add the Teradata JDBC jar in the following location: `<Informatica installation directory>/externaljdbcjars`

Environment Variables

Before you use PowerExchange for Teradata Parallel Transporter API in the native or Hadoop environment, you must configure Java and Teradata environment variables.

In the native environment, set the environment variables on the node that runs the Data Integration Service. In the Hadoop environment, set the environment variables in the `hadoopEnv.properties` file. For more information, see the *Informatica Big Data Management User Guide*.

The following table describes the environment variables that you must configure on Windows and UNIX:

Environment Variable	Value
JAVA_HOME	<Informatica installation directory>/java
JRE_HOME	${JAVA_HOME}/jre
<Shared Library Variable>	Set the shared library variable to include $TWB_ROOT/lib64 and $TD_ICU_DATA/lib64 for 64-bit files. Applicable when you run mappings in the native environment or on the Hive engine without using TDCH.

Set the shared library environment variable based on the operating system. The following table describes the shared library variables for each operating system:

Operating System	Value
Windows	PATH Applicable when you run mappings in the native environment.
Linux	LD_LIBRARY_PATH

The following table describes the environment variables that you must configure on UNIX when you run mappings in the native environment or on the Hive engine without using TDCH:

Environment Variable	Value
COPERR	Set to the location of the errmsg.cat file. For example: /usr/lib
COPLIB	Set to the location of the clispb.dat file. For example: /usr/lib.
THREADONOFF	On UNIX and Linux operating systems, set the THREADONOFF environment variable to 1 to enable multithreading support for Teradata Parallel Transporter processes.
NLSPATH	Set to the following directory: /opt/teradata/client/15.10/msg/%N:/opt/teradata/client/15.10/odbc_64/msg/%N:/usr/lib/nls/msg/%L/%N:/usr/lib/nls/msg/%L/%N.cat

Database Privileges

Before you use PowerExchange for Teradata Parallel Transporter API, verify that you have read and write permissions on the Teradata database and select privileges on specific Data Dictionary tables.

Verify that you have the following database privileges:

- Read permission to read data from Teradata.

- Write permission to write data to Teradata.
- Select privileges on DBC.Tables, DBC.Columns, DBC.UDTInfo, and DBC.Databases.
  For information about select privileges, see the Teradata JDBC Driver documentation.

JDBC Configuration

Before you use PowerExchange for Teradata Parallel Transporter API, configure JDBC on the node where the Data Integration Service runs and on the machine where you installed the Developer tool.

Before you test a Teradata PT connection, perform the following tasks:

- To test a Teradata PT connection from the Administrator tool, copy the Teradata JDBC jars to the following directory on the node where the Data Integration Service runs:
  `<Informatica Installation Directory>\<version folder>\services\shared\jars\thirdparty`
- To test a Teradata PT connection from the machine where you installed the Developer tool, update the CLASSPATH environment variable to include the Teradata JDBC jars on the client machine.
  You must also verify that the CLASSPATH is set to the following directories where you installed the JDBC driver:

  `<JDBC Driver Installation Directory>\Tera-JDBC\tdgssconfig.jar` and `<JDBC Driver Installation Directory>\Tera-JDBC\terajdbc4.jar`

Sqoop Configuration for Running Teradata Mappings on the Blaze Engine

If you run a Teradata mapping on a Hortonworks or Cloudera cluster, and run the mapping on the Blaze engine, the Data Integration Service runs the mapping through Sqoop. If you use a Hortonworks cluster, the Data Integration Service invokes the Hortonworks Connector for Teradata at run time. If you use a Cloudera cluster, the Data Integration Service invokes the Cloudera Connector Powered by Teradata at run time.

To run a Teradata mapping on the Blaze engine with Sqoop, perform the following tasks:

1. Install Informatica.
2. Based on the cluster you use, download the JAR files for Hortonworks Connector for Teradata or Cloudera Connector Powered by Teradata.
3. Define Sqoop arguments in the Teradata PT connection.

Downloading JAR Files for Hortonworks Connector for Teradata or Cloudera Connector Powered by Teradata

To run Teradata mappings on the Blaze engine through Sqoop, you must download the relevant JAR files based on the cluster you use, and copy the JAR files to the node where the Data Integration Service runs. At

run time, the Data Integration Service copies the JAR files to the Hadoop distribution cache so that the JAR files are accessible to all the nodes in the Hadoop cluster.

1. If you use a Hortonworks cluster, perform the following steps:
   a. Click the following URL: http://hortonworks.com/downloads/#addons
   b. Find the Hortonworks Connector for Teradata package. The package has the following naming convention: `hdp-connector-for-teradata-<version>-distro.tar.gz`
   c. Download all the JAR files in the package.
   d. Download the `avro-mapred-1.7.4-hadoop2.jar` file from the following URL: https://archive.apache.org/dist/avro/avro-1.7.4/java/
2. If you use a Cloudera cluster, perform the following steps:
   a. Click the following URL: http://www.cloudera.com/downloads.html
   b. Find the Cloudera Connector Powered by Teradata package. The package has the following naming convention: `sqoop-connector-teradata-<version>.tar.gz`
   c. Download all the JAR files in the package.
   d. Download the `terajdbc4.jar` file and `tdgssconfig.jar` file from the following URL: http://downloads.teradata.com/download/connectivity/jdbc-driver
3. On the node where the Data Integration Service runs, copy all the JAR files mentioned in the earlier steps to the following directory:

   ```
   <Informatica installation directory>\externaljdbcjars
   ```

Defining Sqoop Arguments in the Teradata PT Connection

Define Sqoop arguments in the Teradata PT connection to specify how you want to read and write data.

1. In the Developer tool, click **Window** > **Preferences**.
2. Click **Informatica** > **Connections**.
3. Expand the domain.
4. Select **Databases** > **Teradata PT** and click the Teradata PT connection that you want to use to run the mapping on the Blaze engine.
5. Click **Edit**.
6. In the **Additional Sqoop Arguments** field, enter the arguments that Sqoop must use to process the data. For example, enter `--method split.by.amp`. Separate multiple arguments with a space.

   See the Hortonworks for Teradata Connector and Cloudera Connector Powered by Teradata documentation for a list of arguments that you can specify.

   Note: If you use Hortonworks for Teradata Connector, the --split-by argument is required if you add two or more source tables in the read operation. If you use Cloudera Connector Powered by Teradata, the --split-by argument is required in the source connection if the source table does not have a primary key defined.
7. Click **Test Connection** to verify that you can connect to the Teradata database.

   The properties for data access are validated at run time.
8. Click **Finish**.

Teradata Connector for Hadoop Configuration for the Hive Engine

When you run a Teradata mapping on the Hive engine, you can use the Teradata Connector for Hadoop (TDCH) Command Line Edition to increase performance. TDCH is a set of API and tools that Teradata Corporation provides for parallel processing of data between Teradata databases and the Hadoop ecosystem of products. You can download TDCH from the Teradata Developer Exchange website.

When you run a Teradata mapping on the Hive engine, by default, the Data Integration Service pushes the mapping to a Hadoop cluster and processes the mapping with one mapper task. You can enable TDCH to run a Teradata mapping on the Hive engine. TDCH uses multiple mapper tasks to read and write the data, which significantly increases the performance.

TDCH uses a configuration file named `InfaTDCHConfig.txt` to read and write data. You can define configuration options in the `InfaTDCHConfig.txt` file to specify how you want TDCH to read and write the data.

To run a Teradata mapping on the Hive Engine with TDCH, perform the following tasks:

1. Verify the TDCH prerequisites.
2. Install and configure TDCH.
3. Configure the `InfaTDCHConfig.txt` file.
4. Configure the EnableTdch custom property.

Configuration Parameters for Teradata Connector for Hadoop

Configuration parameters determine how TDCH reads data from and writes data to Teradata.

When you run a Teradata mapping on the Hive engine, you can define the following configuration parameters in the `InfaTDCHConfig.txt` file:

libjars

Set the libjars parameter to the paths that contain the TDCH JAR files.

method

Set the method parameter to define the method that TDCH must use to write data to the Teradata database.

You can set the method to batch.insert or internal.fastload.

nummappers

Set the nummappers parameter to define the maximum number of mapper tasks that TDCH must use while writing data. You can enter an integer value that is greater than or equal to zero. If you enter zero, the number of mapper tasks will be the same as the number of file blocks in HDFS.

Use either this parameter or the numreducers parameter, but not both.

read_nummappers

Set the read_nummappers parameter to define the maximum number of mapper tasks that TDCH must use while reading data. You can enter an integer value that is greater than or equal to zero. If you enter zero, the number of mapper tasks will be the same as the number of file blocks in HDFS.

read_method

Set the read_method parameter to define the method that TDCH must use to read data from the Teradata database.

You can set the read_method parameter to one of the following values:

- split.by.hash
- split.by.value
- split.by.partition
- split.by.amp

For more information about the configuration options, see the Teradata Connector for Hadoop documentation.

Verifying the Teradata Connector for Hadoop Prerequisites

Before you install and configure TDCH and run a mapping on the Hive engine, perform the following tasks:

1. Verify that the Data Integration Service user has the MapR ticket available on all the nodes of the Hadoop cluster.
2. Create a user account in the Teradata Developer Exchange website so that you can download the TDCH installation package from the website.
3. To read data by using TDCH, verify that the following conditions are met:
   - The cluster contains the following folder in the HDFS file system: `/user/yarn/`

     Note: If the folder does not exist, you must create it.
   - If the owner of the `/user/yarn/` folder is not yarn, the user who runs the mapping has rwxrwxrwx permissions on the `/user/yarn/` folder.
4. Ensure that the Hive Client is configured on all the nodes of the Hadoop cluster.
   To verify, run the `hive` command from each node.
5. If the Hadoop cluster is enabled for Kerberos, ensure that the Data Integration Service user has a valid Kerberos ticket on all the nodes of the cluster. If the ticket is not valid, the mapping fails. To generate a valid ticket, run the `kinit` command on each node of the cluster.

Installing and Configuring Teradata Connector for Hadoop

Install and configure TDCH to run a Teradata mapping on the Hive engine.

1. Go to the Teradata Developer Exchange website: http://developer.teradata.com/
2. Download the Teradata Connector for Hadoop Command Line Edition installation package that is relevant for the Hadoop distribution that you use. For example, `teradata-connector-1.4.3-hadoop2.x.noarch.rpm`.
3. Place the `teradata-connector.rpm` on all the nodes of the Hadoop cluster, and install the TDCH installation package.

   The rpm package installs TDCH in the following directory: `/usr/lib/tdch/<version number folder>`
4. Access the following directory on the machine where you installed the Informatica services:

   `<Informatica installation directory>\services\shared\hadoop\<Hadoop Distribution Name>\infaConf\`
5. Find the file named `hadoopEnv.properties`.

6. Make the following changes in the `hadoopEnv.properties` file:
   a. Set the HADOOP_CLASSPATH to include the paths of the TDCH JAR files and the `hive-site.xml` file as follows:

      ```
      infapdo.env.entry.hadoop_classpath=HADOOP_CLASSPATH=<Path of the dependent TDCH JAR files and the directory where the hive-site.xml file is stored>
      ```

      For example: `infapdo.env.entry.hadoop_classpath=HADOOP_CLASSPATH=/opt/cloudera/parcels/CDH/lib/hive/conf:/opt/cloudera/parcels/CDH/lib/hive/lib/*:/usr/lib/tdch/1.5/lib/*:$HADOOP_CLASSPATH`
   b. Set the INFA_MAPRED_CLASSPATH variable to include the paths of the TDCH JAR files and the `hive-site.xml` file.

      For example, set the INFA_MAPRED_CLASSPATH as follows:
      `infapdo.env.entry.mapred_classpath=INFA_MAPRED_CLASSPATH=<Path of the dependent TDCH JAR files and the directory where the hive-site.xml file is stored>`

Configuring the InfaTDCHConfig.txt File

Configure the `InfaTDCHConfig.txt` file to define the parameters that TDCH must use to read and write data when you run a Teradata mapping on the Hive engine. Informatica ships a sample `InfaTDCHConfig.txt` file. Customize the sample file based on your requirements.

1. Access the following directory:

   ```
   <Informatica installation directory>\services\shared\hadoop\<Hadoop_Distribution_Directory>\infaConf
   ```
2. Find the file named `InfaTDCHConfig.txt`.

   Back up the file before you modify it.

   Note: If the `InfaTDCHConfig.txt` file is missing from the `<Informatica installation directory>\services\shared\hadoop\biginsights_3.0.0.1\infaConf` directory, you must manually copy the file from another Hadoop distribution directory.
3. Use a text editor to open the file and edit the properties.
4. Set the libjars parameter to the paths that contain the dependent jars required for TDCH. Perform the following tasks:
   a. To get the list of dependent jars required to run a TDCH job, open the readme file from the following location: `/usr/lib/tdch/1.5`
   b. Under **Dependencies**, view the **Hive Job** section to find the list of jar names.
   c. Locate the jars on the Hadoop nodes in the following directory: `hive/lib`.

      For example: On the Cloudera nodes, verify if the jars are available in the following directory:

      ```
      /opt/cloudera/parcels/CDH-5.8.4-1.cdh5.8.4.p0.5/lib/hive/lib/
      ```

      Note: Ensure that all the jars are available even though the version of the readme and the jar file might differ.
   d. Add all the jars along with the PATH to the libjars variable in the `InfaTDCHConfig.txt` file.

      For example:

      ```
      libjars=/opt/cloudera/parcels/CDH/lib/hive/lib/libthrift-0.9.3.jar,
      /opt/cloudera/parcels/CDH/lib/hive/lib/commons-pool-1.5.4.jar,
      /opt/cloudera/parcels/CDH/lib/hive/lib/commons-dbcp-1.4.jar,
      /opt/cloudera/parcels/CDH/lib/hive/lib/libfb303-0.9.3.jar,
      ```

```
/opt/cloudera/parcels/CDH/lib/hive/lib/jdo-api-3.0.1.jar,
/opt/cloudera/parcels/CDH/lib/hive/lib/datanucleus-rdbms-3.2.9.jar,
/opt/cloudera/parcels/CDH/lib/hive/lib/datanucleus-core-3.2.10.jar,
/opt/cloudera/parcels/CDH/lib/hive/lib/datanucleus-api-jdo-3.2.6.jar,
/opt/cloudera/parcels/CDH/lib/hive/lib/antlr-runtime-3.4.jar,
/opt/cloudera/parcels/CDH/lib/hive/lib/hive-cli-1.1.0-cdh5.8.4.jar,
/opt/cloudera/parcels/CDH/lib/hive/lib/hive-exec-1.1.0-cdh5.8.4.jar,
/opt/cloudera/parcels/CDH/lib/hive/lib/hive-jdbc-1.1.0-cdh5.8.4.jar,
/opt/cloudera/parcels/CDH/lib/hive/lib/hive-metastore-1.1.0-cdh5.8.4.jar
```

5. Set other parameters based on how you want TDCH to process the data.
6. Save the file with the name `InfaTDCHConfig.txt`.

Configuring the EnableTdch Custom Property

You must configure the EnableTdch custom property to enable TDCH to run a Teradata mapping on the Hive engine.

1. Log in to the Informatica Administrator as an Administrator.
2. Select the Data Integration Service in the Domain Navigator to open the service properties.
3. In the contents panel, click the **Processes** view.
4. Click **Edit** in the **Custom Properties** section.

   The **Edit Custom Properties** dialog box appears.
5. Click **New**.

   The **New Custom Property** dialog box appears.
6. Enter the custom property name as `EnableTdch` and the value as `True`.
7. Click **OK** twice.
8. In the contents panel, click the **Properties** view.
9. Click **Recycle Service**.

   The **Recycle Service** dialog box appears.
10. Choose to wait until all processes complete or to abort all processes.
11. Click **OK**.

The EnableTdch custom property is enabled at the Data Integration Service level.

Enable Support for Lookup Transformations with Teradata Data Objects

To use Lookup transformations with a Teradata data object in Hadoop pushdown mode, you must copy the Teradata JDBC drivers to the Informatica installation directory.

You can download the Teradata JDBC drivers from Teradata. For more information about the drivers, see the following Teradata website: http://downloads.teradata.com/download/connectivity/jdbc-driver.

The software available for download at the referenced links belongs to a third party or third parties, not Informatica LLC. The download links are subject to the possibility of errors, omissions or change. Informatica assumes no responsibility for such links and/or such software, disclaims all warranties, either express or implied, including but not limited to, implied warranties of merchantability, fitness for a particular purpose, title and non-infringement, and disclaims all liability relating thereto.

Copy the `tdgssconfig.jar` and `terajdbc4.jar` files from the Teradata JDBC drivers to the following directory on the machine where the Data Integration runs and every node in the Hadoop cluster: `<Informatica installation directory>/externaljdbcjars`

Additionally, you must copy the `tdgssconfig.jar` and `terajdbc4.jar` files to the following directory on the machine where the Developer tool runs: `<Informatica installation directory>\clients\externaljdbcjars`.

CHAPTER 3

PowerExchange for Teradata Parallel Transporter Connections

This chapter includes the following topics:

Teradata Parallel Transporter Connection Overview

Use a Teradata PT connection to access Teradata tables.

Create a connection to import Teradata table metadata, create data objects, and run mappings. The Developer tool uses the connection when you import a data object. The Data Integration Service uses the connection when you run mappings.

Use the Developer tool, Administrator tool, or infacmd to create a Teradata PT connection.

Teradata Parallel Transporter Connection Properties

Use a Teradata PT connection to access Teradata tables. The Teradata PT connection is a database type connection. You can create and manage a Teradata PT connection in the Administrator tool or the Developer tool.

Note: The order of the connection properties might vary depending on the tool where you view them.

The following table describes Teradata PT connection properties:

Property	Description
Name	Name of the connection. The name is not case sensitive and must be unique within the domain. The name cannot exceed 128 characters, contain spaces, or contain the following special characters: `~ ` ! $ % ^ & * ( ) - + = { [}] | \ : ; " ' < , > . ? /`
ID	String that the Data Integration Service uses to identify the connection. The ID is not case sensitive. It must be 255 characters or less and must be unique in the domain. You cannot change this property after you create the connection. Default value is the connection name.
Description	Description of the connection. The description cannot exceed 765 characters.
Location	Domain where you want to create the connection.
Type	Connection type. Select Teradata PT.
User Name	Teradata database user name with the appropriate read and write permissions to access the database.
Password	Password for the Teradata database user name.
Driver Name	Name of the Teradata JDBC driver.
Connection String	Connection string used to access metadata from the database. Use the following connection string: `jdbc:teradata://<hostname>/database=<database name>,tmode=ANSI,charset=UTF8`

The following table describes the properties for data access:

Property	Description
TDPID	Name or IP address of the Teradata database machine.
Database Name	Teradata database name. If you do not enter a database name, Teradata PT API uses the default login database name.
Data Code Page	Code page associated with the database. When you run a mapping that writes data to a Teradata target, the code page of the Teradata PT connection must be the same as the code page of the Teradata target. Default is UTF-8.
Tenacity	Number of hours that Teradata PT API continues trying to log on when the maximum number of operations run on the Teradata database. Must be a positive, non-zero integer. Default is 4.
Max Sessions	Maximum number of sessions that Teradata PT API establishes with the Teradata database. Must be a positive, non-zero integer. Default is 4.
Min Sessions	Minimum number of Teradata PT API sessions required for the Teradata PT API job to continue. Must be a positive integer between 1 and the **Max Sessions** value. Default is 1.

Property	Description
Sleep	Number of minutes that Teradata PT API pauses before it retries to log on when the maximum number of operations run on the Teradata database. Must be a positive, non-zero integer. Default is 6.
Use Metadata JDBC URL for TDCH	Indicates that the Teradata Connector for Hadoop (TDCH) must use the JDBC URL that you specified in the connection string under the metadata access properties. Default is selected. Clear this option to enter a different JDBC URL that TDCH must use when it runs the mapping.
TDCH JDBC Url	Enter the JDBC URL that TDCH must use when it runs a Teradata mapping. Use the following format: `jdbc:teradata://<hostname>/database=<database name>,tmode=ANSI,charset=UTF8` This field is available only when you clear the **Use Metadata JDBC URL for TDCH** option.
Data Encryption	Enables full security encryption of SQL requests, responses, and data on Windows. Default is disabled.
Additional Sqoop Arguments	This property is applicable if you use a Hortonworks or Cloudera cluster, and run a Teradata mapping on the Blaze engine through Sqoop. Enter the arguments that Sqoop must use to process the data. For example, enter `--method split.by.amp`. Separate multiple arguments with a space. See the Hortonworks for Teradata Connector and Cloudera Connector Powered by Teradata documentation for a list of arguments that you can specify. **Note:** If you use Hortonworks for Teradata Connector, the --split-by argument is required if you add two or more source tables in the read operation. If you use Cloudera Connector Powered by Teradata, the --split-by argument is required in the source connection if the source table does not have a primary key defined.
Authentication Type	Method to authenticate the user. Select one of the following authentication types: - Native. Authenticates your user name and password against the Teradata database specified in the connection. - LDAP. Authenticates user credentials against the external LDAP directory service. Default is Native.

Creating a Teradata Parallel Transporter Connection

Before you import Teradata data objects or run mappings, create a Teradata PT connection in the Developer tool.

1. Click **Window** > **Preferences**.
2. Click **Informatica** > **Connections**.
3. Expand the domain.
4. Select **Databases** > **Teradata PT** and click **Add**.
5. Enter a connection name.
6. Enter an ID for the connection.

7. Optionally, enter a connection description.
8. Select the domain where you want to create the connection.
9. Select the type of connection.
10. Click **Next**.
11. Configure the connection properties for metadata access and data access.
12. Click **Test Connection** to verify that you can connect to the Teradata database.
    The properties for data access are validated at run time.
13. Click **Finish**.

CHAPTER 4

PowerExchange for Teradata Parallel Transporter API Data Objects

This chapter includes the following topics:

Teradata Data Object Overview

A Teradata data object is a physical data object that uses Teradata as a source and a target. A Teradata data object is the representation of data that is based on a Teradata table. You can configure the data object read and write operation properties that determine how data can be read from or loaded to Teradata tables.

Import the Teradata table into the Developer tool to create a Teradata data object. Create a data object read or write operation for the Teradata data object. Then, you can add the data object read or write operation to a mapping.

Teradata Data Object Sections

The Teradata data object contains sections to edit the object name and the properties.

After you create a Teradata data object, you can change the data object properties in the following data object views:

- **Overview** view. Edit the Teradata data object name, description, and tables.
- **Data Object Operation** view. Edit the properties that the Data Integration Service uses when it reads data from or writes data to a Teradata table.

When you create a mapping for a Teradata read or write operation, you can view the data object properties in the **Properties** section.

Teradata Data Object Overview Properties

The Teradata **Overview** section displays general information about the Teradata data object and detailed information about the Teradata table that you imported.

The following table describes the general properties that you configure for a Teradata data object:

Property	Description
Name	Name of the Teradata data object.
Description	Description of the Teradata data object.
Connection	Name of the Teradata PT connection. Click **Browse** to select another Teradata PT connection.

The following table describes the properties of the Teradata table that you import:

Property	Description
Name	Name of the Teradata table.
Type	Native data type of the Teradata table.
Description	Description of the Teradata table.

Teradata Data Object Read Operation Properties

The Data Integration Service reads data from a Teradata table based on the data object read operation properties. The Developer tool displays the data object read operation properties for the Teradata data object in the **Data Object Operation** section.

You can view or configure the data object read operation from the input and target properties.

- Input properties. Represents data that the Data Integration Service passes into the mapping pipeline. Select the input properties to edit the port properties and specify the advanced properties of the data object read operation.
- Source properties. Represents data that the Data Integration Service reads from the Teradata table. Select the source properties to view data such as the name and description of the Teradata table. Create key columns to identify each row in a data object read operation, and create key relationships between data object read operations.

Input Properties of the Data Object Read Operation

The input properties represent data that the Data Integration Service passes into the mapping pipeline. Select the input properties to edit the port properties of the data object read operation. You can also specify advanced data object read operation properties to read data from Teradata tables.

The input properties of the data object read operation include general properties that apply to the data object read operation. They also include port, source, and advanced properties that apply to the data object read operation.

You can edit the input properties of the data object read operation from the **Ports** and **Advanced** tabs.

Sources Properties

The sources properties list the Teradata tables in the data object read operation.

Advanced Properties

The advanced properties allow you to specify data object read operation properties to import data from Teradata tables.

The following table describes the advanced properties that you configure in the data object read operation:

Property	Description
Pre SQL	SQL command that the Data Integration Service runs against the source database before it reads from the source.
Post SQL	SQL command that the Data Integration Service runs against the source database before it writes to the target.
Select Distinct	Specifies if you want to select only unique rows. The Data Integration Service includes a SELECT DISTINCT statement if you choose this option.

Property	Description
Select SQL Statement	This property is applicable in the following scenarios: - You run a Teradata mapping on a Hortonworks cluster and on the Blaze engine. - You run a Teradata mapping on a Cloudera cluster and on the Blaze engine. - You run a Teradata mapping in the native environment. Enter an SQL statement to override the default query. You can specify the columns that you want to use from the source database. You can also parameterize the SQL statement.
Query Band Expression	The query band expression to be passed to the Teradata PT API. A query band expression is a set of name-value pairs that identify a query's originating source. In the expression, each name-value pair is separated by a semicolon and the expression ends with a semicolon. For example, ApplicationName=Informatica;Version=9.5.0;ClientUser=A;.
Source Table Name	Table name from which you want to extract rows.
Source Table Schema	Schema name from which you want to extract rows.
Driver Tracing Level	Determines Teradata PT API tracing at the driver level: - TD_OFF. Teradata PT API disables tracing. - TD_OPER. Teradata PT API enables tracing for driver-specific activities for Teradata. - TD_OPER_ALL. Teradata PT API enables all driver-level tracing. - TD_OPER_CLI. Teradata PT API enables tracing for activites involving CLIv2. - TD_OPER_NOTIFY. Teradata PT API enables tracing for activities involving the Notify feature. - TD_OPER_OPCOMMON. Teradata PT API enables tracing for activities involving the operator common library. Default is TD_OFF. The input locale for the value of the Driver Tracing Level attribute is in the English language only.
Infrastructure Tracing Level	Determines Teradata PT API tracing at the infrastructure level. Select one of the following values: - TD_OFF. Teradata PT API disables tracing. - TD_OPER. Teradata PT API enables tracing for driver-specific activities for Teradata. - TD_OPER_ALL. Teradata PT API enables all driver-level tracing. - TD_OPER_CLI. Teradata PT API enables tracing for activities involving CLIv2. - TD_OPER_NOTIFY. Teradata PT API enables tracing for activities involving the Notify feature. - TD_OPER_OPCOMMON. Teradata PT API enables tracing for activities involving the operator common library. Default is TD_OFF. You must enable the driver tracing level before you can enable the infrastructure tracing level. The input locale for the value of the Infrastructure Tracing Level attribute is in the English language only.

Property	Description
Trace File Name	File name and path of the Teradata PT API trace file. Default path is `$<Informatica installation directory>\tomcat\bin`. Default file name is `<Name of the TPT Operator>_timestamp`. For example, `LOAD_20091221`.
Spool Mode	Determines the spool mode Teradata PT API uses to extract data from Teradata. You can choose one of the following spool modes: - Spool. Teradata PT API spools data while extracting data from Teradata. - NoSpool. Teradata PT API does not spool data while extracting data from Teradata. If the database does not support the NoSpool option, Teradata PT API uses the Spool option. - NoSpoolOnly. Teradata PT API does not spool while extracting data from Teradata. Default is Spool. The input locale for the value of the Spool Mode attribute is in the English language only.

Note: If you set the custom property EnableTdch to true, the advanced properties in the data object read operation are not applicable.

If you run a mapping on a Hortonworks or Cloudera cluster and on the Blaze engine, the following properties are supported for Teradata data objects and customized data objects:

- Select SQL Statement
- Source Table Name
- Source Filter

If you run a mapping on a Hortonworks cluster and on the Blaze engine, the Source Table Prefix property is supported for Teradata data objects and customized data objects.

Source Properties of the Data Object Read Operation

The source properties represent the data that is populated based on the Teradata tables that you added when you created the data object. The source properties of the data object read operation include general and column properties that apply to the Teradata tables.

You can edit the source properties of the data object read operation from the **Keys** and **Column** tabs.

Keys Properties

Displays the Teradata table that has a primary key.

You can view the name and description of the table that has a primary key. The **Available Fields** section displays the columns that are not part of the primary key. The **Selected Fields** section displays the columns that are part of the primary key.

Column Properties

The column properties display the data types, precision, and scale of the target property in the data object read operation.

The following table describes the column properties of the data object read operation:

Property	Description
Name	Name of the column property.
Type	Native data type of the column property.
Precision	Maximum number of significant digits for numeric data types, or maximum number of characters for string data types. For numeric data types, precision includes scale.
Scale	Maximum number of digits after the decimal point for numeric values.
Primary Key	Determines if the column property is a part of the primary key.
Description	Description of the column property.

Teradata Data Object Write Operation Properties

The Data Integration Service writes data to a Teradata table based on the data object write operation properties. The Developer tool displays the data object write operation properties for the Teradata data object in the **Data Object Operation** section.

You can view or configure the data object write operation from the input and target properties.

- Input properties. Represents data that the Data Integration Service passes into the mapping pipeline. Select the input properties to edit the port properties and specify the advanced properties of the data object write operation.
- Target properties. Represents data that the Data Integration Service writes to the Teradata table. Select the target properties to view data such as the name and description of the Teradata table, create key columns to identify each row in a data object write operation, and create key relationships between data object write operations.

Input Properties of the Data Object Write Operation

The input properties represent data that the Data Integration Service passes into the mapping pipeline. Select the input properties to edit the port properties of the data object write operation. You can also specify advanced data object write operation properties to load data into Teradata tables.

The input properties of the data object write operation include general properties that apply to the data object write operation. They also include port, source, and advanced properties that apply to the data object write operation.

You can edit the input properties of the data object write operation from the **Ports** and **Advanced** tabs.

Ports Properties

The input ports properties list the data types, precision, and scale of the data object write operation.

The following table describes the input ports properties that you configure in a data object write operation:

Property	Description
Name	Name of the port property.
Type	Data type of the port property.
Precision	Maximum number of significant digits for numeric data types, or maximum number of characters for string data types. For numeric data types, precision includes scale.
Scale	Maximum number of digits after the decimal point for numeric values.
Description	Description of the port property.

Advanced Properties

The advanced properties allow you to specify data object write operation properties to load data into Teradata tables.

The following table describes the advanced properties that you configure in the data object write operation:

Property	Description
Macro Database	Name of the database that stores the macros Teradata PT API creates when you select the Stream system operator. The Stream system operator uses macros to modify tables. It creates macros before Teradata PT API begins loading data and removes them from the database after Teradata PT API loads all rows to the target. If you do not specify a macro database, Teradata PT API stores the macros in the log database.
Truncate Table	Teradata PT API deletes all rows in the Teradata target before it loads data. This attribute is available for the Stream, Update, and Load system operators. Default is disabled.
Pause Acquisition	Causes the load operation to pause before writing data to the Teradata PT API target. Default is disabled.
Query Band Expression	Query band expression to be passed to Teradata PT API. A query band expression is a set of name-value pairs that identify a query's originating source. In the expression, each name-value pair is separated by a semicolon and the expression ends with a semicolon. For example, ApplicationName=Informatica;Version=9.5.0;ClientUser=A;.

Property	Description
Mark Duplicate Rows	Specifies how Teradata PT API handles duplicate rows when it tries to insert or upsert rows in the target table. Select one of the following values: - None. If Teradata PT API receives a row marked for insert or upsert that causes a duplicate row in the target table, Teradata PT API does not write the row to the error table and does not mark it as an error row in the workflow log. - For Insert. If Teradata PT API receives a row marked for insert that exists in the target table, Teradata PT API writes the row to the error table and marks it as an error row in the workflow log. - For Update. If Teradata PT API receives a row marked for update that causes a duplicate row in the target table, Teradata PT API writes the row to the error table and marks it as an error row in the workflow log. - Both. If Teradata PT API receives a row marked for insert or upsert that causes a duplicate row in the target table, Teradata PT API writes the row to the error table and marks it as an error row in the workflow log. Default is For Insert. The input locale for the value of the Mark Duplicate Rows attribute is in the English language only.
Log Database	Name of the database that stores the log tables.
Log Table Name	Name of the restart log table.
Error Database	Name of the database that stores the error tables.
Error TableName1	Name of the first error table. Use this attribute to override the default error table name.
Error TableName2	Name of the second error table. Use this attribute to override the default error table name.
Drop Log/Error	Drops the existing log and work tables when you run a mapping. This attribute is available for a Teradata target ODBC connection. Default is disabled.
Serialize	Uses the Teradata PT API serialize mechanism to reduce locking overhead when you select the Stream system operator. Default is enabled.
Pack	Number of statements to pack into a request when you select the Stream system operator. Must be a positive, nonzero integer. Default is 20. Minimum is 1. Maximum is 600.
Pack Maximum	Causes Teradata PT API to determine the maximum number of statements to pack into a request when you select the Stream system operator. Default is disabled.
Buffers	Determines the maximum number of request buffers that may be allocated for the Teradata PT API job when you select the Stream system operator. Must be a positive, nonzero integer. Default is 3. Minimum is 2.

Property	Description
Error Limit	Maximum number of records that can be stored in the error table before Teradata PT API terminates the Stream system operator job. Must be -1 or a positive, nonzero integer. Default is -1, which specifies an unlimited number of records.
Replication Override	Specifies how Teradata PT API overrides the normal replication services controls for an active Teradata PT API mapping. Select one of the following values: - On. Teradata PT API overrides normal replication services controls for the active mapping. - Off. Teradata PT API disables override of normal replication services for the active mapping when change data capture is active. - None. Teradata PT API does not send an override request to the Teradata Database. Default is None. The input locale for the value of the Replication Override attribute is in the English language only.
Driver Tracing Level	Determines Teradata PT API tracing at the driver level. Select one of the following values: - TD_OFF. Teradata PT API disables tracing. - TD_OPER. Teradata PT API enables tracing for driver-specific activities for Teradata. - TD_OPER_ALL. Teradata PT API enables all driver-level tracing. - TD_OPER_CLI. Teradata PT API enables tracing for activities involving CLIv2. - TD_OPER_NOTIFY. Teradata PT API enables tracing for activities involving the Notify feature. - TD_OPER_OPCOMMON. Teradata PT API enables tracing for activities involving the operator common library. Default is TD_OFF. The input locale for the value of the Driver Tracing Level attribute is in the English language only.
Infrastructure Tracing Level	Determines Teradata PT API tracing at the infrastructure level. Select one of the following values: - TD_OFF. Teradata PT API disables tracing. - TD_OPER. Teradata PT API enables tracing for driver-specific activities for Teradata. - TD_OPER_ALL. Teradata PT API enables all driver-level tracing. - TD_OPER_CLI. Teradata PT API enables tracing for activities involving CLIv2. - TD_OPER_NOTIFY. Teradata PT API enables tracing for activities involving the Notify feature. - TD_OPER_OPCOMMON. Teradata PT API enables tracing for activities involving the operator common library. Default is TD_OFF. You must enable the driver tracing level before you can enable the infrastructure tracing level. The input locale for the value of the Infrastructure Tracing Level attribute is in the English language only.
Trace File Name	File name and path of the Teradata PT API trace file. Default path is `$<Informatica installation directory>\tomcat\bin`. Default file name is `<Name of the TPT Operator>_timestamp`. For example, `LOAD_20091221`.

Property	Description
System Operator	Specifies the Teradata PT API operator type. Select one of the following values: - Load. Performs bulk loading into an empty Teradata database table. You can use the Load system operator in native and Hadoop environments. - Stream. Performs insert, update, upsert, and delete operations against Teradata database tables in near real-time mode. You can use the Stream system operator in native and Hadoop environments. - Update. Performs insert, update, upsert, and delete operations against Teradata database tables. You can use the Update system operator in native and Hadoop environments. Default is Stream. The input locale for the value of the System Operator attribute is in the English language only.
Mark Missing Rows	Specifies how Teradata PT API handles rows that do not exist in the target table when it tries to update or delete rows. Select one of the following values: - None. If Teradata PT API receives a row marked for update or delete but it is missing in the target table, Teradata PT API does not write the row to the error table and does not mark it as an error row in the workflow log. - For Update. If Teradata PT API receives a row marked for update but it is missing in the target table, Teradata PT API writes the row to the error table and marks it as an error row in the workflow log. - For Delete. If Teradata PT API receives a row marked for delete but it is missing in the target table, Teradata PT API writes the row to the error table and marks it as an error row in the workflow log. - Both. If Teradata PT API receives a row marked for update or delete but it is missing in the target table, Teradata PT API writes the row to the error table and marks it as an error row in the workflow log. Default is None. The input locale for the value of the Mark Missing Rows attribute is in the English language only.
Mark Extra Rows	Specifies how Teradata PT API marks error rows when it tries to update or delete multiple target table rows. Select one of the following values: - None. If Teradata PT API receives a row marked for update or delete that affects multiple rows in the target table, Teradata PT API does not write the row to the error table and does not mark it as an error row in the workflow log. - For Update. If Teradata PT API receives a row marked for update that affects multiple rows in the target table, Teradata PT API writes the row to the error table and marks it as an error row in the workflow log. - For Delete. If Teradata PT API receives a row marked for delete that affects multiple rows in the target table, Teradata PT API writes the row to the error table and marks it as an error row in the workflow log. - Both. If Teradata PT API receives a row marked for update or delete that affects multiple rows in the target table, Teradata PT API writes the row to the error table and marks it as an error row in the workflow log. Default is Both. This attribute is available only for the Stream system operator. The input locale for the value of the Mark Extra Rows attribute is in the English language only.
Insert	Teradata PT API creates a DML group to insert rows. If you do not want to insert rows, clear this option to increase session performance. Default is selected.

Property	Description
Update	Teradata PT API creates a DML group to update rows. Default is not selected.
Upsert	Teradata PT API creates a DML group to update or insert rows. Default is not selected.
Delete	Teradata PT API creates a DML group to delete rows. Default is not selected.
WorkTable Database Name	Name of the database that stores the work tables. Use this attribute to override the default work table database name. If you do not specify a database name, the Data Integration Service uses the target table database.
WorkTable Name	Name of the work tables when you select the Update system operator. The Teradata database creates one work table for each target table. If you do not specify a work table name, the Data Integration Service uses the name <work_table_database>.INFA<number>_WT. The exact table name appears in the session log.
Pre SQL	Pre-session SQL commands to run against the source database before the Data Integration Service reads the source.
Post SQL	Post-session SQL commands to run against the source database after the Data Integration Service writes to the target.
Create or Replace Table at run time	Creates or replaces the Teradata target at run time when you run a dynamic mapping. Select this option if you want the Data Integration Service to create or replace the Teradata target. Default is not selected.

Rules and Guidelines for Advanced Properties

Consider the following rules and guidelines when you configure the advanced properties:

- If you set the custom property EnableTdch to true, the advanced properties in the data object write operation are not applicable.
- If you run the mapping on a Hortonworks or Cloudera cluster and on the Blaze engine, only the **Truncate Table** property is supported.
- When you enable the **Create or Replace Table at Run time** property and run a dynamic mapping, the Integration Service drops the target table and recreates it. When you rerun a mapping and you want to retain the target table that resulted from the first mapping, disable the **Create or Replace Table at Run time** property before you run the mapping.
- When you create a target using dynamic mapping, you do not get the primary key information from the source.
- When you create a target using dynamic mapping, the Data Integration Service creates all target fields as nullable fields even if the source fields are not null.

Also, the Integration Service converts native data types from the source to the following data types in the target:

Source Data Type	Target Data Type
byteint	integer
smallint	integer
float	double
char	nvarchar
nchar	nvarchar
date	timestamp
time	timestamp
boolean	numeric

Target Properties of the Data Object Write Operation

The target properties represent the data that is populated based on the Teradata tables that you added when you created the data object. The target properties of the data object write operation include general and column properties that apply to the Teradata tables.

You can edit the target properties of the data object write operation from the **Keys** and **Column** tabs.

Keys Properties

Displays the Teradata table that has a primary key.

You can view the name and description of the table that has a primary key. The **Available Fields** section displays the columns that are not part of the primary key. The **Selected Fields** section displays the columns that are part of the primary key.

Column Properties

The column properties display the data types, precision, and scale of the target property in the data object write operation.

The following table describes the target column properties of the data object write operation:

Property	Description
Name	Name of the column property.
Type	Native data type of the column property.
Precision	Maximum number of significant digits for numeric data types, or maximum number of characters for string data types. For numeric data types, precision includes scale.

Property	Description
Scale	Maximum number of digits after the decimal point for numeric values.
Primary Key	Determines if the column property is a part of the primary key.
Description	Description of the column property.

Importing a Teradata Data Object

Import a Teradata data object to add to a mapping.

Configure a Teradata PT connection, before you import a Teradata data object.

1. Select a project or folder in the **Object Explorer** view.
2. Click **File** > **New** > **Data Object**.
3. Select **Teradata Data Object** and click **Next**.

   The **New Teradata Data Object** dialog box appears.
4. Enter a name for the data object.
5. Click **Browse** next to the **Location** option and select the target project or folder.
6. Click **Browse** next to the **Connection** option and select the Teradata PT connection from which you want to import the Teradata table metadata.
7. To add a table, click **Add** next to the **Selected Resources** option.

   The **Add Resource** dialog box appears.
8. Select a table. You can search for it or navigate to it.
   - Navigate to the Teradata table that you want to import and click **OK**.
   - To search for a table, enter the name or the description of the table you want to add. Click **OK**.
9. If required, add more tables to the Teradata data object.

   You can also add tables to a Teradata data object after you create it.
10. Click **Finish**.

The data object appears under Data Objects in the project or folder in the **Object Explorer** view.

Creating a Teradata Data Object Read Operation

You can create the data object read operation for one or more Teradata data objects. You can add a Teradata data object read operation to a mapping as a source.

Before you create a Teradata data object read operation, you must create a Teradata data object with at least one Teradata table.

1. Select the data object in the Object Explorer view.
2. Right-click and select **New** > **Data Object Operation**.

The **Data Object Operation** dialog box appears.

3. Enter a name for the data object operation.
4. Select **Read** as the type of data object operation.
5. Click **Add**.

   The **Select Resources** dialog box appears.
6. Select the Teradata table for which you want to create the data object read operation and click **OK**.

   You can select only one data object to a data object read operation.
7. Click **Finish**.

The Developer tool creates the data object read operation for the selected data object.

Creating a Teradata Data Object Write Operation

You can create the data object write operation for one or more Teradata data objects. You can add a Teradata data object write operation to a mapping as a target.

Before you create a Teradata data object write operation, you must create the Teradata data object with at least one Teradata table.

1. Select the data object in the Object Explorer view.
2. Right-click and select **New** > **Data Object Operation**.

   The **Data Object Operation** dialog box appears.
3. Enter a name for the data object operation.
4. Select **Write** as the type of data object operation.
5. Click **Add**.

   The **Select Resources** dialog box appears.
6. Select the Teradata table for which you want to create the data object write operation and click **OK**.

   You can select only one data object to a data object write operation.
7. Click **Finish**.

The Developer tool creates the data object write operation for the selected data object.

CHAPTER 5

PowerExchange for Teradata Parallel Transporter API Mappings

This chapter includes the following topics:

Teradata Mappings Overview

After you create a Teradata data object read or write operation, you can create a mapping.

To read data from a Teradata source, add the Teradata data object read operation as the Read transformation in a mapping. To write data to a Teradata target, add the Teradata data object write operation as the Write transformation in a mapping. You can also configure dynamic mappings with Teradata sources and targets.

Validate and run the mapping. You can also add the mapping to a Mapping task in a workflow and run the workflow.

Hadoop Validation and Run-time Environments

You can configure the mappings to run in the native or Hadoop run-time environment. When you run mappings in the native environment, the Data Integration Service processes the mapping. When you run mappings in the Hadoop environment, the Data Integration Service can push mappings that are developed in the Developer tool to a Hadoop cluster.

For more information about the Hadoop environment, see the *Informatica Big Data Management User Guide*.

Teradata Mapping Example

Your organization, ABC Airlines, needs to analyze flight information from the log files that are generated for each flight to find the number of flights to a particular destination in a month. Create a mapping that reads the flight log information of all the flights and writes specific information to Teradata tables.

You can use the following objects in a Teradata mapping:

Mapping input

The mapping source is a table in an Oracle database or a flat file that contains the flight log information of ABC Airlines.

Import the Oracle or flat file data object with the flight log information. Create a data object read operation for the Oracle data object you have created. Add the data object read operation to the mapping.

Transformation

Use the Filter transformation to specify the month and destination.

Mapping output

The mapping output is a data object write operation to load the data to a Teradata table. Run the mapping either in the native or Hadoop environment.

When you run the mapping, the Data Integration Services writes the extracted information about the number of flights in a month to a specific destination to the Teradata target table. You can then use the information for data analysis.

CHAPTER 6

Teradata Parallel Transporter API Run-time Processing

This chapter includes the following topics:

Teradata Run-time Processing Overview

When you create a Teradata mapping, you define data object operation read or write properties that determine how data is read from Teradata tables or loaded to Teradata targets.

You can set the following data object operation read or write properties:

- System operator
- Error limits
- Names of error tables and log tables
- Strategy that Teradata PT API must use to treat duplicate rows, missing rows, and extra rows when it loads data to a Teradata target

System Operators

The Data Integration Service uses Teradata PT API to connect to Teradata. It loads data by using the Teradata PT API Stream, Load, or Update system operators.

Stream System Operator

You can use the Stream system operator to perform high-speed parallel inserts to empty or existing Teradata tables without locking target tables. You can also perform update, upsert, and delete operations after you define a primary key in the target table.

You can maintain current and accurate data for immediate decision making. The Stream system operator uses macros to modify tables. The Stream system operator creates macros before Teradata PT API begins loading data and removes them from the database after Teradata PT API loads all rows to the target. If you do not specify a macro database, Teradata PT API stores the macros in the log database. You can use the Stream system operator in native and Hadoop environments.

Error Limit for the Stream System Operator

Teradata PT API terminates the Stream system operator job after it reaches the maximum number of records that can be stored in the error table.

The error limit is approximate because the Stream operator sends multiple rows of data at a time to the Teradata table. By the time Teradata PT API processes the message that indicates that the error limit has been exceeded, it might have loaded more records into the error table than the number specified in the error limit.

By default, the error limit value is unlimited. The error limit applies to each instance of the Stream system operator.

Load System Operator

The Load system operator loads a large volume of data at high speed into an empty Teradata table. The Load operator is often used to load initial data to Teradata tables as it inserts data in to individual rows of a target table. You can use the Load system operator in native and Hadoop environments.

Update System Operator

You can use the Update system operator to perform insert, update, upsert, and delete operations against Teradata database tables. You can load data to empty or existing Teradata tables.

If you want to run an update, upsert, or delete operation on a Teradata target table, you must ensure that the target has a primary key column.

You can use the Update system operator in native and Hadoop environments.

Error and Log Tables

Use the log and error tables to view log, error, and rejected data when a mapping runs.

When you use Teradata PT API to run a mapping that loads data to Teradata, Teradata PT API creates log and error tables. The log tables store Teradata PT API restart and log information. The error tables log Teradata errors and rejected data when a mapping runs.

Error Tables

Teradata writes rejected data to error tables ErrorTable1 and ErrorTable2.

ErrorTable1 contains data that was rejected for the following reasons:

- Data conversion errors
- Constraint violations
- Access Module Processor configuration changes

ErrorTable2 contains data that was rejected for the following reasons:

- Unique primary index constraint violations
- Load driver job acquisition phase errors

When you configure a Teradata mapping, you can enter a name for each error table. You can create the error tables in any database. By default, the error tables are stored in the default database. Choose where you want to create the error tables when you configure a mapping to load to Teradata.

When a mapping fails, see the error tables for more information about the errors. Before you run the mapping again, drop the error tables or enter different table names in the data object operation write properties. Otherwise, the mapping fails.

The following table describes the data object operation write properties that allow you to specify error table names:

Property	Description
Error Database	Name of the database that stores the error tables. If you do not enter an error database name in the data object operation write properties, Teradata PT API stores the error tables in the database you specify in the connection object.
Error Table Name1	Name of the first error table. If you do not specify a name for the first error table, the Data Integration Service uses the name *<error_database>.INFA_ET1_<number>*. The table name appears in the mapping log.
Error Table Name2	Name of the second error table. If you do not specify a name for the second error table, the Data Integration Service uses the name *<error_database>.INFA_ET2_<number>*. The table name appears in the mapping log.

Log Tables

When you configure a Teradata mapping, you can enter a log table name. You can create the error tables in any database. By default, the log tables are stored in the default database.

When a mapping fails, see the log table for more information. Before you run the mapping again, drop the log table or enter a different table name in the data object operation write properties. Otherwise, the mapping fails.

The following table describes the data object operation write properties that allow you to specify log table information:

Property	Description
Log Database	Name of the database that stores the log tables. If you do not enter a log database name in the data object operation write properties, Teradata PT API stores the log tables in the database you specify in the connection object.
Log Table Name	Name of the log table. If you do not specify a log table name, the Data Integration Service uses the name *<log_database>.INFA_LT_<number>*. The table name appears in the mapping log.

Row-level Error Processing

When you write data to a Teradata target, you can configure how Teradata PT API treats duplicate rows, missing rows, and extra rows.

Duplicate Rows

You can configure how Teradata PT API handles duplicate rows when it tries to insert or upsert rows in the target table.

You can select one of the following values:

- None. If Teradata PT API receives a row marked for insert or upsert that causes a duplicate row in the target table, Teradata PT API does not write the row to the error table and does not mark it as an error row in the workflow log.
- For Insert. If Teradata PT API receives a row marked for insert that exists in the target table, Teradata PT API writes the row to the error table and marks it as an error row in the workflow log.
- For Update. If Teradata PT API receives a row marked for update that causes a duplicate row in the target table, Teradata PT API writes the row to the error table and marks it as an error row in the workflow log.
- Both. If Teradata PT API receives a row marked for insert or upsert that causes a duplicate row in the target table, Teradata PT API writes the row to the error table and marks it as an error row in the workflow log.

Missing Rows

You can configure how Teradata PT API handles rows that do not exist in the target table when it tries to update or delete rows.

You can select one of the following values:

- None. If Teradata PT API receives a row marked for update or delete but it is missing in the target table, Teradata PT API does not write the row to the error table and does not mark it as an error row in the workflow log.
- For Update. If Teradata PT API receives a row marked for update but it is missing in the target table, Teradata PT API writes the row to the error table and marks it as an error row in the workflow log.
- For Delete. If Teradata PT API receives a row marked for delete but it is missing in the target table, Teradata PT API writes the row to the error table and marks it as an error row in the workflow log.
- Both. If Teradata PT API receives a row marked for update or delete but it is missing in the target table, Teradata PT API writes the row to the error table and marks it as an error row in the workflow log.

Extra Rows

You can configure how Teradata PT API marks error rows when it tries to update or delete multiple target table rows.

You can select one of the following values:

- None. If Teradata PT API receives a row marked for update or delete that affects multiple rows in the target table, Teradata PT API does not write the row to the error table and does not mark it as an error row in the workflow log.
- For Update. If Teradata PT API receives a row marked for update that affects multiple rows in the target table, Teradata PT API writes the row to the error table and marks it as an error row in the workflow log.
- For Delete. If Teradata PT API receives a row marked for delete that affects multiple rows in the target table, Teradata PT API writes the row to the error table and marks it as an error row in the workflow log.
- Both. If Teradata PT API receives a row marked for update or delete that affects multiple rows in the target table, Teradata PT API writes the row to the error table and marks it as an error row in the workflow log.

This attribute is available only for the Stream system operator.

Parameterization

You can parameterize Teradata data object operation properties to override the read and write data object operation properties during run time.

In bulk mode, you can parameterize the following data object read operation properties for Teradata sources:

- Pre SQL
- Post SQL
- Select Distinct
- Select SQL Statement
- Query Band Expression
- Source Table Name
- Source Table Schema
- Driver Tracing Level
- Infrastructure Tracing Level
- Trace File Name
- Spool Mode

In bulk mode, you can parameterize the following data object write operation properties for Teradata targets:

- Macro Database
- Truncate Table
- Pause Acquisition
- Query Band Expression
- Mark Duplicate Rows
- Log Database

- Log Table Name
- Error Database
- Error Table Name1
- Error Table Name2
- Drop Log/Error
- Serialize
- Pack
- Pack Minumum
- Buffers
- Error Limit
- Replication Override
- Driver Tracing Level
- Infrastructure Tracing Level
- Trace File Name
- System Operator
- Mark MISSING Rows
- Mark EXTRA Rows
- Insert
- Update
- Upsert
- Delete
- WorkTable Database Name
- WorkTable Name
- Pre SQL
- Post SQL

The following attributes support partial parameterization:

- Query Band Expression
- Trace File Name

Partitioning

You can configure partitioning for Teradata mappings that you run in the native environment. When a mapping that is enabled for partitioning contains a Teradata data object as a target, the Data Integration Service can use multiple threads to write to the target.

You can configure dynamic partitioning for Teradata Parallel Transporter API data objects. You can configure the partition information so that the Data Integration Service determines the number of partitions to create at run time.

To enable partitioning, administrators and developers perform the following tasks:

Administrators set maximum parallelism for the Data Integration Service to a value greater than 1 in the Administrator tool.

Maximum parallelism determines the maximum number of parallel threads that process a single pipeline stage. Administrators increase the **Maximum Parallelism** property value based on the number of CPUs available on the nodes where mappings run.

Developers can set a maximum parallelism value for a mapping in the Developer tool.

By default, the **Maximum Parallelism** property for each mapping is set to Auto. Each mapping uses the maximum parallelism value defined for the Data Integration Service.

Developers can change the maximum parallelism value in the mapping run-time properties to define a maximum value for a particular mapping. When maximum parallelism is set to different integer values for the Data Integration Service and the mapping, the Data Integration Service uses the minimum value of the two.

Note: When you create a Teradata write data object, ensure that the **Serialize** option is set to **Off** to enable partitioning for the Stream operator.

Dynamic Partitioning

If the volume of data grows or you add more processors, you can adjust partitioning so that the run time does not increase. When you use dynamic partitioning, you can configure the partition information so that the Data Integration Service determines the number of partitions to create at run time.

Some transformations do not support partitioning. When a mapping enabled for partitioning contains a transformation that does not support partitioning, the Data Integration Service uses one thread to run the transformation. The Data Integration Service can use multiple threads to run the remaining mapping pipeline stages.

You can optimize write performance by using dynamic partitioning.

APPENDIX A

Data Type Reference

This appendix includes the following topics:

Data Type Reference Overview

Informatica Developer uses the following data types in Teradata mappings:

- Teradata native data types. Teradata data types appear in the data object operation column properties.
- Transformation data types. Set of data types that appear in the transformations. They are internal data types based on ANSI SQL-92 generic data types, which the Data Integration Service uses to move data across platforms. Transformation data types appear in all transformations in a mapping.

When the Data Integration Service writes to a target, it converts the transformation data types to the comparable native data types.

Teradata Data Types and Transformation Data Types

The following table lists the Teradata data types that the Data Integration Service supports and the corresponding transformation data types:

Teradata Data Type	Range	Transformation Data Type	Range
Bigint	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	Bigint	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 Precision 19, scale 0
Byte	1 to 64,000 bytes	Binary	1 to 104,857,600 bytes
Byteint	-128 to 127	Small Integer	Precision 5, scale 0
Char	1 to 64,000 bytes	String	1 to 104,857,600 characters

Teradata Data Type	Range	Transformation Data Type	Range
Date	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. Precision 19, scale 0	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)
Decimal	Precision 1 to 18, scale 0 to 18	Decimal	Precision 1 to 28, scale 0 to 28
Float	-2.226E+308 to 1.797E+308	Double	Precision 15
Integer	-2,147,483,648 to 2,147,483,647	Integer	-2,147,483,648 to 2,147,483,647 Precision 10, scale 0
Smallint	-32768 to 32768	Small Integer	Precision 5, scale 0
Time	00:00:00.000000 to 23:59:61.999999 Precision 15, scale 6	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)
Timestamp	1 to 19 characters Precision 19 to 26, scale 0 to 6	Date/Time	Jan 1, 0001 A.D. to Dec 31, 9999 A.D. (precision to the nanosecond)
Varbyte	1 to 64,000 bytes	Binary	1 to 104,857,600 bytes
Varchar	1 to 64,000 bytes	String	1 to 104,857,600 characters

When you use Teradata mappings in a Blaze environment, you must convert Decimal, Date, Time, or Timestamp data types for the Teradata object to String data type in the Developer Tool. Byte and Varbyte data types are not supported in the Blaze environment.

INDEX