



Teradata® Data Mover

User Guide

Release 17.20




2022-09-26

Copyright and Trademarks

Copyright © 2009 - 2022 by Teradata. All Rights Reserved.

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

Product Safety

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

Third-Party Materials

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

Warranty Disclaimer

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

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

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

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

Machine-Assisted Translation

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

Feedback

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

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

Confidential Information

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

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

Contents

Chapter 1: Overview	6
Welcome to Teradata Data Mover User Guide	6
Introduction to Teradata Data Mover	7
Supported Copy Methods between Database Software Versions	8
Overview of Teradata Data Mover Components	8
Data Mover Jobs	17
Required Privileges	17
Restrictions	19
Multi-Byte Object Support	22
QueryGrid Support	22
Cloud Staging Copy Service (CS2) and Cloud Staging Area Support	23
Single Sign-on Support	25
Data Dictionary Views	25
Chapter 2: Data Mover Command Line Interface	27
Command-Line Interface Overview	27
Data Mover XML Files	33
Data Mover Setup and Configuration	41
Job Management	73
DSA Incremental Copy Jobs	86
DSA Cloud Staging Copy Jobs	92
Working with Staging Databases	97
About Copying Objects	108
About Copying Entire Databases	160
Chapter 3: Data Mover Portlets	172
Data Mover Setup	172
Data Mover	185
Chapter 4: Data Mover RESTful API	217
Data Mover RESTful API	217
Supported Data Mover Commands for RESTful API	218
Data Mover RESTful Service	220
Viewing Data Mover REST Samples	220
Swagger User Interface	221
Chapter 5: Monitoring Teradata Data Mover	222
Using Teradata Ecosystem Manager to Monitor Jobs	222
Using Data Mover to Monitor Status	227

Chapter 6: Performance Monitoring and Tuning	228
Monitoring Job Performance	228
Data Mover Best Practices	228
About Tuning Data Mover	228
Adding a Data Mover Server to the Viewpoint Monitoring Portlet	233
Chapter 7: High Availability	235
Automatic Failover	235
The Synchronization Service	243
Chapter 8: Troubleshooting	253
Solutions for Job Problems	253
Logging Levels	254
Failure Scenarios	255
Using Server Management Logging	256
System Health	261
Creating a Diagnostic Bundle for Support	262
User Cannot Choose Data Mover Daemon in the Data Mover Portlet	264
Job is Queued Even Though No Other Work is Executing	264
Appendix A: Data Mover Command Parameters	265
Appendix B: RESTful API	374
Appendix C: Data Mover XML Schemas	496
Appendix D: Additional Information	641

Overview

Welcome to Teradata Data Mover User Guide

Using Teradata Data Mover User Guide

Using Data Mover, you can define and edit jobs that copy specified database objects, such as tables, users, views, macros, stored procedures, and statistics, from one Teradata database system to another Teradata database system.

This guide explains the Data Mover component capabilities, usage, and interconnection in detail. Troubleshooting and command parameters are also available for user support.

This guide is for:

- Database administrators
- System administrators
- Software developers, production users, and testers

Why Would I Use this Content?

This guide has the elaborated basic information about Data Mover:

- Purpose and structure
- Supported copy methods
- Restrictions
- Command-Line interface
- Portlet familiarization
- RESTful API
- Automatic failover and synchronization
- Troubleshooting

How Do I Use This Content?

Depending upon your roles and requirements, you can refer to this guide. To use this guide, you need to be familiar with:

- Dual-active systems
- Teradata Database
- Teradata system hardware

How Do I Get Started?

Start with the [Overview](#) section to understand the basics of Data Mover and proceed to the following topics as per your requirements.

- [Data Mover Command Line Interface](#)
- [Data Mover Portlets](#)
- [Data Mover RESTful API](#)
- [Monitoring Teradata Data Mover](#)
- [Performance Monitoring and Tuning](#)
- [High Availability](#)
- [Troubleshooting](#)
- [Data Mover Command Parameters](#)
- [Data Mover XML Schemas](#)

References to Other Relevant Content

- For information on Teradata connector limitations, refer to *Teradata® QueryGrid™ Installation and User Guide*, B035-5991.
- For detailed information about the query band feature using valid key-value pair, refer to *Teradata® Database SQL Data Definition Language - Syntax and Examples*, B035-1144.

Introduction to Teradata Data Mover

Data Mover enables you to define and edit jobs that copy specified database objects, such as tables, users, views, macros, stored procedures, and statistics, from one Teradata Database system to another Teradata Database system. Tables and data returned by views can be copied between systems. View definitions cannot be copied.

In addition to tables and statistics, the following database objects associated with tables can be copied with the tables, but not by themselves:

- Hash indexes
- Join indexes
- Triggers

Dual-active support of active copying between two systems is provided through the graphical, REST, and command-line interfaces of Data Mover, which allows you to:

- Establish systems to enable copying, or recover systems to re-enable copying.
- Copy a large volume of data from one Teradata Database system to another on a regularly-scheduled basis.
- Perform a partial copy of tables from a source Teradata Database system to a target system.

In the command-line interface, you can use commands to define jobs, run jobs, and edit jobs. If Teradata Viewpoint is installed in your environment, you can access the functionality of Data Mover in a user-friendly graphic interface.

RESTful API is available with Data Mover and supports all job related commands using the REST API. For more information, see [Data Mover RESTful API](#).

If Teradata Ecosystem Manager is installed in your environment, it can be used to monitor and control Data Mover.

Note:

For information about objects supported between databases, see [Objects Supported During Moves Between Databases](#).

Supported Copy Methods between Database Software Versions

For Data Mover dependencies and compatibility information, see the [Teradata® Data Mover Compatibility Matrix](#).

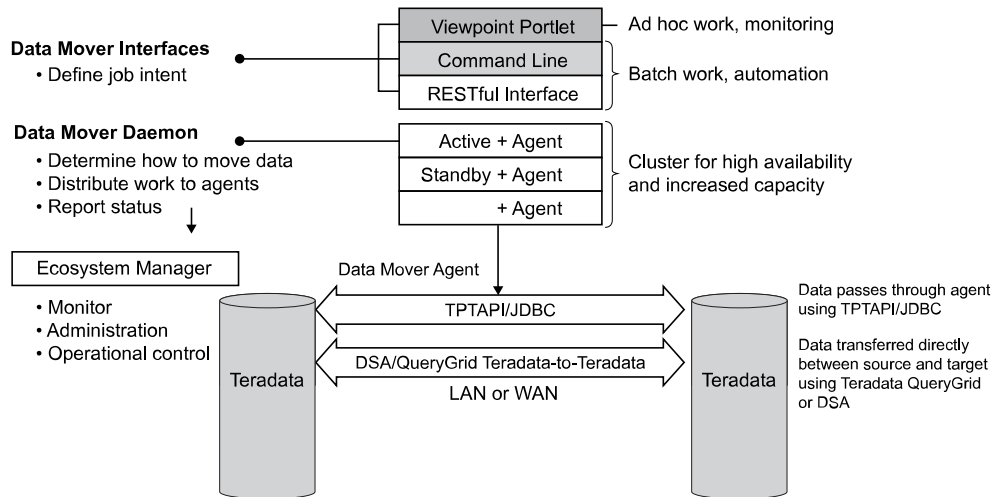
Overview of Teradata Data Mover Components

The next figure shows the main components of Data Mover:

- The Data Mover daemon
- The Data Mover agent
- The user interfaces:
 - Graphic (Data Mover portlet)
 - Command-line
 - RESTful API

When specifying job actions using one of the interfaces, the Data Mover daemon generates job steps. The Data Mover daemon then distributes the tasks for each job step among the Data Mover agents, which run the job tasks. Data Mover is set up to report job status to Teradata Ecosystem Manager, which can monitor and control Data Mover jobs.

Data Mover Components



Data Mover Deployment Options

Data Mover can be deployed using the following platforms:

- Teradata Multi-Purpose Server (TMS)
- Virtual machine on a Consolidated Teradata Multi-Purpose Server (CTMS)
- Teradata Cloud (previously IntelliCloud)
- Private Cloud on virtual machine
- Amazon Web Services (AWS)
- Microsoft Azure (Azure)
- Google Cloud

Data Mover Data Transfer Options

Data Mover supports data transfer between the following systems:

From System	To System
Vantage on-premises (IntelliFlex)	Vantage on-premises (IntelliFlex)
Vantage in the Cloud (public or private)	Vantage in the Cloud (public or private)
Vantage on-premises (IntelliFlex)	Vantage in the Cloud (public or private)
Vantage in the Cloud (public or private)	Vantage on-premises (IntelliFlex)
Teradata Cloud	Vantage in the Cloud (public or private)
Vantage in the Cloud (public or private)	Teradata Cloud

Data Mover Daemon

The main component of Teradata® Data Mover is the Data Mover daemon, which can do the following:

- Processes incoming requests from one of the Data Mover interfaces
- Queries Teradata Database systems for information on specified objects
- Generates a plan to copy objects from the source system to the target system, based on job specifications
- Distributes work for each job to the Data Mover agents
- Captures and stores job information such as the job plan and output generated by each job instance
- Sends status information to Teradata Ecosystem Manager and Server Management

The Data Mover daemon uses a scaled-down version of the Teradata Database to store job information and metadata. All communication between the Data Mover daemon and the Data Mover command-line interface, Data Mover agents, or other external components is through the Java Message Service (JMS) bus. All messages between these components are in XML format.

One task of the Data Mover daemon is to send job status information to Teradata Ecosystem Manager and Server Management, enabling critical failures to be reported to Teradata immediately. The Data Mover Daemon generates Teradata Ecosystem Manager event information for jobs launched with the portlet and command-line interfaces, then sends the status to Teradata Ecosystem Manager if that product is present and the user has configured Data Mover to do so.

Properties File for the Data Mover Daemon

The `daemon.properties` file contains configuration properties for the Data Mover daemon.

```
# Copyright (C) 2009-2021 by Teradata Corporation.
# All Rights Reserved.
# TERADATA CORPORATION CONFIDENTIAL AND TRADE SECRET
#-----
-----
# File: "daemon.properties"
#
# Purpose: This file contains all of the properties used by the DM Daemon.
#
# Caution: Do not modify the property names/values in this file unless absolutely
sure
# of the implications to the DM Daemon.
#
# Note: Any line of text in this document that begins with a '#' character is a
comment and
# has no affect on the DM Daemon. However, comments should not be modified.
#
# All properties under LOGGING comment are used for logging purposes
#
```

```

#-----
-----

# Purpose: The hostname or IP address of the machine running the
# Java Message Service (JMS) Message Broker.
# Default: localhost
# Other examples include:
# broker.url=10.0.1.199
# broker.url=hostname
# broker.url=[fe80::114:23ff:fe1d:32fb]
broker.url=localhost

# Purpose: The port number on the machine in which the
# Java Message Service (JMS) Message Broker is listening.
# Default: 61616
broker.port=61616

# Purpose: When set to true, a connection to a
# Secondary Java Message Service (JMS) Message Broker can be
# established in case the Primary Java Message Service (JMS) Broker fails.
# Default: false
cluster.enabled=false

# Purpose: The message time to live in milliseconds; zero is unlimited
# Be SURE to synchronize clocks of all dm hosts so messages do not expire
unintentionally
# causing failed or hanging requests to Daemon.
# Default: 1 hr
jms.response.timetolive=3600000

# Purpose: A long-lived server port on the machine running the DM Daemon, which is
used for inbound
# socket connections from DM Agents.
# Default: 25168

# Purpose: When set to true, Fatal Error messages can be sent to TVI
# Default: true
tvi.useLogger=true

# Purpose: Scripts are used to run to collect TVI diagnostic bundles file.
# DO NOT change this directory location.
tvi.diagnosticbundle.script.dir=/opt/teradata/datamover/support/diagnosticbundle

# Purpose: TVI diagnostic bundle files are saved to this directory

```

```

# This value need to be the same as SUPPORT_BUNDLE_DIR in /etc/opt/teradata/sm3g.
sm3gnode.conf
tvi.diagnosticbundle.dir=/var/opt/teradata/datamover/support/diagnosticbundle

# Purpose: The maximum number of jobs allowed to run on the daemon at the same
time.
# Additional jobs are placed on the queue and executed when slots become
available
# Default: 20
jobExecutionCoordinator.maxConcurrentJobs=20

# Purpose: The maximum number of jobs allowed in the job queue.
# Additional jobs are placed in a higher level memory queue until slots are
available in the job queue.
# Default: 20
jobExecutionCoordinator.maxQueuedJobs=20

# Purpose: The hostname or IP address for the ViewPoint Authentication server.
# Default: http://localhost
viewpoint.url=http://dm-vp1

# Purpose: The port number for the ViewPoint Authentication server.
# Default: 80
viewpoint.port=80

# Purpose: The hostname and IP address for the Querygrid Manager servers.
# Support clustered QueryGrid managers, could have upto 2 urls, separate by
comma.
# e.g querygrid.manager.urls=https://host1:9443,https://host2:9443
# Default: https://localhost:9443
querygrid.manager.urls=https://localhost:9443

#-----LOGGING-----

# Purpose: Set Logging level to info. User has 6 options.
# From most amount of logging to least: trace < debug < info < warn < error <
fatal
rootLogger.level=info

# Purpose: Informs the logging application to use a specific appender and it's
properties. DO NOT CHANGE
appender.rolling.type=RollingFile
appender.rolling.name=RollingFile
appender.rolling.layout.type=PatternLayout

```

```

appender.rolling.layout.pattern=%d [%t] %-5p %c{3}(%L) - %m%n
appender.rolling.policies.type=Policies
appender.rolling.policies.size.type=SizeBasedTriggeringPolicy
appender.rolling.strategy.type=DefaultRolloverStrategy
logger.rolling.name=com.teradata
logger.rolling.appenderRef.rolling.ref=RollingFile

# Purpose: Allow's user the ability to change the location of the log file.
# If changing log file location, please give the absolute path of the file;
# for example, /var/log/dmAgent.log
# for windows os: use slash instead of back slash:
# For Example: C:/Program File/Teradata/Log/dmDaemon.log
appender.rolling.fileName=/var/opt/teradata/datamover/logs/dmDaemon.log
appender.rolling.filePattern=/var/opt/teradata/datamover/logs/dmDaemon.log.%i

# Purpose: The max size of the logging file before being rolled over to backup
files.
appender.rolling.policies.size.size=20MB
# Purpose: The number of backup logging files created, after the max number, the
oldest file is erased.
appender.rolling.strategy.max=5

service_user=dmsuser

# Purpose: Data Mover Rest Interface for DSA job status notification
dm.rest.endpoint=https://localhost:1443/datamover

# Purpose: DSA Rest Interface for DSA job status notification
dsa.rest.endpoint=https://localhost:9090/dsa
#-----

```

Data Mover Agent

The Data Mover daemon uses Data Mover agents to perform the work for each Data Mover job; therefore, at least one agent is required. Multiple agents can run tasks in parallel, improving performance. To maximize performance, Teradata recommends that each agent resides on its own server, separate from the Data Mover daemon server. However, it is not a requirement. A Data Mover agent can run on the same server as the Data Mover daemon.

The Data Mover daemon splits work into tasks, which are placed on a JMS queue for the next available Data Mover agent. The Data Mover agent uses one of the following utilities to perform the task, then listens on the queue for its next task:

- Data Stream Architecture (DSA)
- Teradata Parallel Transporter (TPT) API

- Teradata JDBC Driver
- QueryGrid

Properties File for the Data Mover Agent

The agent.properties file contains configuration properties for the Data Mover agent.

```
# Copyright (C) 2009-2021 by Teradata Corporation.
# All Rights Reserved.
# TERADATA CORPORATION CONFIDENTIAL AND TRADE SECRET
#-----
-----
# File: "agent.properties"
#
# Purpose: This file contains all of the properties used by the DM Agent.
#
# Caution: Do not modify the property names/values in this file unless absolutely
sure
# of the implications to the DM Agent.
#
# Note: Any line of text in this document that begins with a '#' character is a
comment and
# has no affect on the DM Agent. However, comments should not be modified.
#
# All properties under LOGGING Comment are used for logging purposes
#
#-----
-----

# Purpose: The Agent Identifier
# Default: Agent1
agent.id=Agent1

# Purpose: The hostname or IP address of the machine running the
# Java Message Service (JMS) Message Broker.
# Default: localhost
broker.url=localhost

# Purpose: The port number on the machine in which the
# Java Message Service (JMS) Message Broker is listening.
# Default: 61616
broker.port=61616

# Purpose: When set to true, a connection to a
# Secondary Java Message Service (JMS) Message Broker can be
```

```

# established in case the Primary Java Message Service (JMS) Broker fails.
# Default: false
cluster.enabled=false

# Purpose: Port that will be used by ARC
# Default: 25268

# Purpose: The maximum number of tasks allowed to run on this agent at the same
time.
# This property has the side-effect of reducing parallelism among multiple Agents
if
# set too high, because one Agent will grab all the tasks on the queue
# Default: 5
agent.maxConcurrentTasks=5

#When set to true, Fatal Error messages can be sent to TVI
tvi.useLogger=true

# Purpose: Scripts are used to run to collect TVI diagnostic bundles file.
# DO NOT change this directory location.
tvi.diagnosticbundle.script.dir=/opt/teradata/datamover/support/diagnosticbundle

# Purpose: TVI diagnostic bundle files are saved to this directory
# This value need to be the same as SUPPORT_BUNDLE_DIR in /etc/opt/teradata/sm3g.
sm3gnode.conf
tvi.diagnosticbundle.dir=/var/opt/teradata/datamover/support/diagnosticbundle

#-----LOGGING-----

# Purpose: Set Logging level to info.  User has 6 options.
# From most amount of logging to least: trace < debug < info < warn < error <
fatal
rootLogger.level=info

# Purpose: Informs the logging application to use a specific appender and it's
properties.  DO NOT CHANGE
appender.rolling.type=RollingFile
appender.rolling.name=RollingFile
appender.rolling.layout.type=PatternLayout
appender.rolling.layout.pattern=%d [%t] %-5p %c{3}(%L) - %m%n
appender.rolling.policies.type=Policies
appender.rolling.policies.size.type=SizeBasedTriggeringPolicy
appender.rolling.strategy.type=DefaultRolloverStrategy
logger.rolling.name=com.teradata

```

```

logger.rolling.appenderRef.rolling.ref=RollingFile

# Purpose: Allow's user the ability to change the location of the log file.
# If changing log file location, please give the absolute path of the file;
# for example, /var/log/dmAgent.log
# for windows os: use slash instead of back slash:
# For Example: C:/Program File/Teradata/Log/dmAgent.log
appender.rolling.fileName=/var/opt/teradata/datamover/logs/dmAgent.log
appender.rolling.filePattern=/var/opt/teradata/datamover/logs/dmAgent.log.%i

# Purpose: The max size of the logging file before being rolled over to backup
files.
appender.rolling.policies.size.size=10MB
# Purpose: The number of backup logging files created, after the max number, the
oldest file is erased.
appender.rolling.strategy.max=3

service_user=dmsuser
#-----

```

Data Mover Portlet

The Data Mover portlet allows you to set up, launch, edit, and monitor jobs that copy database objects between Teradata Database systems in a graphic-user interface. Because the interface for the Data Mover portlet is user-friendly and intuitive, you can accomplish copy tasks easily with Data Mover by browsing objects to copy in the database hierarchy of the source system and creating a job definition. The portlet eliminates the need to edit XML parameter files or remember the syntax of common commands. To use the Data Mover portlet, you must have Teradata Viewpoint installed.

Command-Line Interface

The command-line interface provides setup commands that enable you to set up and configure the Data Mover system, including the daemon and agents and job operation commands that enable you to create, run, monitor, update, edit, and delete jobs. The setup commands provide functionality that corresponds to that provided by the **Data Mover Setup** portlet. The job management commands provide functionality that corresponds to that provided by the **Data Mover** portlet. For a list of commands and valid parameters, see [About Data Mover Commands](#).

Each command in the interface requires a set of parameters for operation, which can be listed as part of the command-line arguments or specified in an XML document. If the same parameter is defined on the command line and in the XML file, the command line takes precedence. Multiple instances of the command-line interface are valid.

RESTful API

The RESTful API provides functionality similar to the command-line interface. The RESTful API can be accessed from any third-party or Teradata server using any standard REST client. The APIs provide a programmatic method to create, run, monitor, update, edit, stop, clean up, and delete jobs. For more information, see [Data Mover RESTful API](#).

Data Mover Jobs

A Data Mover job consists of the following:

- A user-provided definition of database objects that Data Mover copies from the source to the target, such as a copy of a table from the PROD1 Teradata Database system to the DEV1 Teradata Database system.
- A programmatically-generated plan of sequential and parallel tasks to optimize job execution, such as distributing job execution into tasks across several servers.

A job is useful for the following tasks:

- Copying database objects.
- Copying one or more tables from an existing Teradata Database system to a backup Teradata Database system, where data in both systems stays synchronized.
- Copying data from a production Teradata Database system to a staging or test Teradata Database system.
- Migrating data regularly in one direction from the original to a backup.

When creating a job, you select source and target databases and specify the source database objects to copy. You can also refine the job definition to control performance, logging, and other factors.

Note:

Spaces in the user or account name for the source or target ID cause jobs to fail.

The jobs continue to exist after being run, including job definition and history, whether successful or not. You can delete these jobs if they are no longer needed.

Note:

Only one instance of a specific job can run at one time.

Data Mover jobs that are created with the command-line or RESTful API `move` command act the same as running the `create` command followed by the `start` command.

Required Privileges

Required `source_user` permissions are listed in the following table:

Permission	Description
SELECT	Required to select data from source table.
DUMP	Required to archive data from source table.
MONITOR	Required to collect TVI asset information (such as, site id, cluster id, and product type).
CREATE TABLE	Required to create source staging tables.
DROP TABLE	Required to drop source staging tables.
INSERT	Required to insert data into source staging tables.

Required *target_user* permissions are listed in the following table:

Note:

Additional grants may be required when performing a full database copy and object types exist in the source database that are not covered by the GRANT statements.

Permission	Description
CREATE TABLE	Required to create staging, target, error, log, and work tables.
DROP TABLE	Required to drop staging, target error, log, and work tables.
CREATE MACRO	Required by Teradata PT API Stream operator to create macro.
DROP MACRO	Required by Teradata PT API Stream operator to drop macro.
EXECUTE MACRO	Required by Teradata PT API Stream operator to run macro.
INSERT	Required to insert data into staging, target, error, log, and work tables.
SELECT	Required to select data from staging table.
RESTORE	Required to restore data to target table.
MONITOR	Required to collect TVI asset information (such as, site id, cluster id, and product type).
DELETE	Required to delete data from the target table.

Required staging database permissions using an existing target table or target database are listed in the following table:

Permission	Description
SELECT WITH GRANT OPTION	Required to select data from source table.
INSERT WITH GRANT OPTION	Required to insert data into target table.
UPDATE WITH GRANT OPTION	Required to update data in target table.

Permission	Description
DELETE WITH GRANT OPTION	Required to delete data from target table.

Required DSA permissions on the SYSBAR database for source and target users are listed in the following table:

Permission	Description
DROP TABLE CREATE TABLE SELECT UPDATE DELETE EXECUTE PROCEDURE CREATE MACRO DROP MACRO	Required for DSMAIN system configuration.

Restrictions

Data Mover supports copying databases, tables, users, hash and join indexes, statistics, triggers, macros, stored procedures, and views. Data Mover also supports other types of tables, including those that have restrictions depending on the utility that is used when copying the object.

Note:

Data Mover does not allow copying of any or all DBC data objects to a different database. DBC data is metadata to be used only when restoring a database. If you define a job to move any data objects from DBC or copy any data objects to DBC, an exception error occurs.

Supported Database Objects

In general, you cannot copy any database objects that use features that exist in newer versions of Teradata Database to older versions of Teradata Database.

Database Object Name	Older to Newer Versions	Between Same Versions	Newer to Older Versions	Restrictions
Databases	Yes	Yes	No	Can copy entire databases with Teradata DSA. See Copying an Entire Database . Cannot copy full databases from newer to older versions of Teradata Database.
Tables	Yes	Yes	Yes	Cannot copy multi-set tables from newer to older systems when target only has 1 AMP or

Database Object Name	Older to Newer Versions	Between Same Versions	Newer to Older Versions	Restrictions
				<p>1 AMP clusters (Teradata PT Update operator restriction).</p> <p>Cannot use TPTAPI_LOAD to copy multi-set tables if the allowTptLoadForMultiset table attribute is not set to true.</p> <p>Full-table copies with tables that have show table output larger than 1MB are not supported. <i>Show table</i> is the text needed to reproduce the table definition. There is no limit to how much data the table can contain.</p> <p>Cannot use Teradata DSA to copy tables from newer to older versions of Teradata Database.</p> <p>Cannot use Teradata DSA if the override_lock_access parameter is set to true.</p> <p>Cannot use Teradata DSA if the source and target TDPIDs are the same.</p> <p>Cannot use DSA to copy partial tables unless source staging is used.</p>
Global Temporary Tables	Yes	Yes	Yes	
Hash and Join Indexes	Yes	Yes	Yes	See Copying Join and Hash Indexes .
Macros	Yes	Yes	Yes	The underlying table must already exist on the target system or be copied with the job.
PPI Tables	Yes	Yes	Yes	
Queue Tables	Yes	Yes	No	Teradata PT Export operator does not support queue tables.
Schema	Yes	Yes	No	<p>Schema is supported in Teradata Database 16.00 or later.</p> <p>Teradata DSA and Teradata JDBC support copying tables with an associated schema object.</p> <p>When copying a table that references a schema object, the schema object must either already exist on the target system or must be copied in the same job.</p> <p>Data Mover does not relocate or rename schema objects.</p> <p>Schemas are created only in the SYSUDTLIB database.</p> <p>If you copy a schema with the same name that exists on the target system and overwrite_existing_objects is set to true, the schema is</p>

Database Object Name	Older to Newer Versions	Between Same Versions	Newer to Older Versions	Restrictions
				overwritten if no columns on the target system reference the schema.
Stored Procedures	Yes	Yes	Yes	The underlying table must already exist on the target system or be copied with the job.
Table Statistics	Yes	Yes	Yes	See About Copying Statistics .
Referencing Tables (parent/child)	Yes	Yes	Yes	See About Copying Tables with Referential Integrity .
Tables with LOB Columns	Yes	Yes	Yes	Teradata JDBC is used for newer to older versions of the Teradata Database. You cannot use Teradata JDBC to move tables with more than 15 LOB columns. Teradata PT API does not support copying data in tables with LOB-based columns.
Tables with Identity Columns	Yes	Yes	Yes	Will not copy from newer to older versions if identity column defined with GENERATED ALWAYS statement.
Tables with User-Defined Type (UDT) Columns	Yes	Yes	Yes	UDT must already be defined on target system. If a job copies a table with UDT columns, the source user and target user must have UDTUSAGE access to SYSUDTLIB to create the job. If a job copies a table with UDT columns and the job uses the Teradata DSA utility, there are no access restrictions for the target user.
Tables with XML, ST_Geometry, JSON Columns	Yes	Yes	Yes	Teradata DSA, and Teradata JDBC support copying tables with XML, ST_Geometry, or JSON column types. Teradata JDBC is used for copying from newer to older versions of the Teradata Database. Teradata PT API does not support copying data with these column types. Due to syntax and format changes in Teradata Database 16.00, compare DLL fails for these column types when copying between systems where one system is earlier than Teradata Database 16.00 and the other is Teradata Database 16.00 or later.
Columnar Tables	Yes	Yes	Yes	Columnar tables with Primary AMP index or Primary Index are supported in Teradata Databases 15.10 or later. Cannot force the use of TPTAPI_LOAD or TPTAPI_UPDATE.

Database Object Name	Older to Newer Versions	Between Same Versions	Newer to Older Versions	Restrictions
Triggers	Yes	Yes	Yes	See About Copying Triggers .
Views	Yes	Yes	Yes	You must use TPTAPI_UPDATE, TPTAPI_STREAM, T2T, or JDBC to copy view data. DSA cannot be used when source staging table is not in use. The TPTAPI_LOAD operator cannot be used.
Foreign Server Objects	Yes	Yes	Yes	You can copy foreign server objects only between Teradata Database 15.00 or later.
Function Alias Objects	Yes	Yes	Yes	You can copy function alias objects only between Teradata Database 16.20 Feature Update 1 or later.
Users	Yes	Yes	No	User must already exist on the target Teradata system. Data Mover moves user objects as database objects. Only the objects within the user are moved. Passwords, permissions, and roles associated with the user are not moved. See About Copying Entire Databases .

Multi-Byte Object Support

In general, Data Mover supports copying tables that contain any kind of character data. Therefore, tables with CHARACTER SET UNICODE, CHARACTER SET LATIN, CHARACTER SET KANJISJIS, or CHARACTER SET KANJI1 columns will be copied correctly in most cases. However, the following scenarios could cause Teradata Database errors when copying data:

- Different export width on the source and target systems
- Copying KANJI1 data to or from a Teradata Database system
- Copying a Character PPI table with Teradata PT API or JDBC when the default collation of the target user does not match the PPI collation of the source table
- Copying a table that contains a mixture of CHARACTER SET UNICODE, CHARACTER SET LATIN, CHARACTER SET KANJISJIS, or CHARACTER SET KANJI1 columns

If a Teradata Database "untranslatable character" error occurs when copying data, the source and target session character sets may need different values in the Data Mover job.

Multi-byte object names are only supported when they are specified in UTF-8.

QueryGrid Support

Data Mover supports copying tables between Teradata systems using the QueryGrid 2.x T2T utility.

Data Mover 17.20 onwards you can choose to use a foreign server for a T2T job from source system and target system.

Restrictions

- The following data types are not supported by T2T: Dataset with CSV storage. For further information on Teradata connector limitations, refer to *Teradata® QueryGrid™ Installation and User Guide*, B035-5991.
- By default, Data Mover allows five concurrent T2T tasks running on one system. This value applies to both the source and target systems. The default value can be modified using the `configuration.xml` file; however, allowing excess amount of T2T tasks to run at the same time causes source and system resource issues.
- The QueryGrid Teradata-to-Teradata connection allows a maximum of 2048 table columns to transfer, row size must be less than 64k.
- Transaction semantics are not supported between systems using T2T.

Known Issues

- A T2T job cannot be stopped while performing a task. Tasks run until completion and the `cleanup` command does not end these tasks. If more than 20 T2T tasks are running concurrently on the same system, including T2T queries not initiated by Data Mover, system resources are affected.
- Data Mover T2T jobs fail with the following error, even though QueryGrid is set up properly on both the source and target systems:

```
Error: T2T cannot be used to move data.
Reason: Cannot use T2T when QueryGrid is not installed either on source or
target system.
```

Data Mover validation fails if the **`databaseQueryService.useBaseViewsOnly` property** is set to **false** and the Teradata users provided for the job do not have access to QueryGrid functions.

Version	Solution
QueryGrid 2.x	Verify that the Teradata source user has SELECT access to the TD_SYSFNLIB.QGINITIATOREXPORT function and that the target user has SELECT access to the TD_SYSFNLIB.QGINITIATORIMPORT function.

Cloud Staging Copy Service (CS2) and Cloud Staging Area Support

The Cloud Staging Copy Service (CS2) is a REST service packaged with Data Mover. The Cloud Staging Copy Service provides a method of copying data from a source system to a target system through cloud storage. The name of this cloud storage is Cloud Staging Area.

Make sure to specify a Cloud Staging Area for using DSA to copy data to or from a Vantage system with Cloud Bridge enabled, or to or from a Vantage Cloud Lake Edition system. Otherwise use QueryGrid, TPT, or JDBC.

Note:

Currently the Data Mover supports only AWS S3 cloud storage.

You can create cloud staging area in following two methods:

- [User Defined Target Groups](#)
- [Cloud Staging Copy Service Defined Target Groups](#)

Following are the general restrictions for a cloud staging area:

- You can have any one type of configuration (either user defined or Cloud Staging Copy Service defined) in one cloud staging area.

Note:

You can have different configurations for different cloud staging areas, but the configuration has to be uniform in individual cloud staging areas.

- You can have only one AWS account per cloud staging area. Multiple cloud staging areas cannot share one AWS account.
- You cannot have duplicate source/target Teradata systems in a source/target pair.

For example, you can have:

```
"source_target_pairs": [
{
  "source_system": "sourceSystemA",
  "target_system": "targetSystemB"
},
,
{
  "source_system": "sourceSystemB",
  "target_system": "sourceSystemA"
}
```

But not:

```
"source_target_pairs": [
{
  "source_system": "sourceSystemA",
  "target_system": "targetSystemB"
},
```



```
{
  "source_system": "sourceSystemC",
  "target_system": "sourceSystemD"
},
{
  "source_system": "sourceSystemA",
  "target_system": "sourceSystemB"
}
```

Single Sign-on Support

If an IdP is integrated with Viewpoint, then Data Mover accepts a JWT token from Viewpoint (through DM Portlets) to Daemon to authenticate Viewpoint users. The Single sign on integration with Data Mover is currently only supported on Cloud platforms.

Data Dictionary Views

The Teradata Database Data Dictionary has tables and views that reside in the system user DBC. The tables are reserved for use by the system and contain information about the data in the system. The views provide access to the information in the tables. Data Dictionary has the following table views available, depending on the version of the Teradata Database in use:

- Base or V Version: better performance
- X or VX Version: increased security, reduced performance

The following table lists which versions of the view Data Mover uses, depending on your configuration settings and Teradata Database version. Depending on the version of Teradata Database in use, the following is true:

- When all four views are available and the `databaseQueryService.useBaseViewsOnly` configuration setting is true (default), the Base or V Version of the view is used.
- When all four views are available and the `databaseQueryService.useBaseViewsOnly` configuration setting is false, the X or VX Version of the view is used.

Data Dictionary View Name	Version of DBC View Used with <code>databaseQueryService. useBaseViewsOnly</code> Setting		Source or Target
	True	False	
DBC.ALL_RI_Children	V	VX	Source only
DBC.Columns	V	VX	Source and Target
DBC.ColumnStats	V	V	Source only
DBC.Databases	V	VX	Source and Target
DBC.DataSetSchemaInfo	V	V	Source only

Data Dictionary View Name	Version of DBC View Used with databaseQueryService. useBaseViewsOnly Setting		Source or Target
	True	False	
DBC.DiskSpace	Base or V	Base or V	Source and Target
DBC.Functions	V	VX	Target only
DBC.IndexStats	V	V	Source only
DBC.Indices	V	VX	Source and Target
DBC.JoinIndices	V	V	Source and Target
DBC.Maps	V	VX	Source and Target
DBC.MultiColumnStats	V	V	Source only
DBC.Server	V	VX	Target only
DBC.Tables	V	VX	Source and Target
DBC.TableSize	V	VX	Source and Target
DBC.TblSrvInfo	V	VX	Target only
DBC.Triggers	V	VX	Source and Target
DBC.RI_Distinct_Children	V	VX	Source and Target
DBC.RI_Distinct_Parents	V	VX	Source and Target
DBC.Children	V	VX	Source only

Note:

- Users that run commands and jobs in Data Mover must have SELECT permissions on the views listed in the table for Data Mover to function properly.
- Data Mover uses the HELP VIEW DBC.Columns output to determine which columns to query for the DBC Columns View.

Data Mover Command Line Interface

Command-Line Interface Overview

Data Mover provides a command-line interface that enables you to carry out many of the same actions that can be carried out in the **Data Mover** and **Data Mover Setup** portlets. The Data Mover command-line interface includes setup commands for configuring Data Mover daemons and agents and for creating, running, editing, and monitoring Data Mover copy jobs.

You can run the commands on an ad-hoc basis, as an alternative to using the portlets, and you can switch between the command-line and portlet interfaces. For example, you can create jobs in the **Data Mover** portlet and then view them in the command-line interface using the `list_jobs` command. Or you can create a job using the command-line interface and an XML file, and then change the job settings for the job using the **Data Mover** portlet. You can also develop scripts to automate Data Mover commands and to use with UNIX cron or other job-scheduling applications. For example, you might want to have a copy job run automatically every night at 1 a.m.

For a complete list of commands and associated parameters, see [About Data Mover Commands](#).

Accessing the Command-Line Interface

After the Data Mover command-line interface package has been successfully installed, the command line can be accessed from the console. You can run Data Mover commands from any file system location without navigating to the installation directory.

1. From any directory, type `datamove` followed by the command and its parameters.

Accessing Data Mover Command-Line Help

You can display the basic syntax structure and view a list and brief description of all of the commands available in the Data Mover command-line interface.

You can also specify a command name to view more detailed information about the command, including a syntax example and a list and description of each of the parameters associated with the command.

1. Type `datamove --help` in the command line, and do one of the following:

Option	Description
View all commands	<ul style="list-style-type: none"> • Press Enter to view a list and brief description of all of the commands.
View a specific command	<ul style="list-style-type: none"> • Add a specific command name to view information about a particular command, including a list of its parameters. • Press Enter.

For example, for information about the create command, type `datamove --help create` and press **Enter**. The following is displayed:

```
Data Mover Command Line 17.06.00.00
Connected to Daemon version 17.06.00.00
NAME:
create - Create Command

DESCRIPTION:
Creates a job on the DM Daemon based on the information from the arguments and the
parameter file

EXAMPLE:
datamove create -job_name job1 -f parameters.xml

Parameters:
Parameter                                Example      Description
job_name                                job1         (optional)Name for the job, must
be unique. Generated if unspecified
job_priority                            MEDIUM      (optional)Execution priority for
job. HIGH/MEDIUM/LOW. Default is MEDIUM.
source_tdpid                            Checks      Source Teradata database
source_user                             TD_API_user (optional)Source Teradata logon id
source_password                         123456789   (optional)Source Teradata logon
password
source_logon_mechanism                  NTLM        (optional)Mechanism to use when
logging onto source system
source_logon_mechanism_data             joe@domain1 (optional)Additional parameters
for the logon mechanism being used
source_account_id                       account_id   (optional)Source Teradata logon
account ID
target_tdpid                            Leo         Target Teradata database
target_user                             TD_API_user (optional)Target Teradata logon id
target_password                         123456789   (optional)Target Teradata logon
password
target_logon_mechanism                  NTLM        (optional)Mechanism to use when logging
onto target system
target_logon_mechanism_data             joe@domain1 (optional)Additional parameters for
the logon mechanism being used
target_account_id                       account_id   (optional)Target Teradata logon account ID
data_streams                           4           (optional)Number of data streams to
use
source_sessions                         4           (optional)Number of session on the
source
target_sessions                         4           (optional)Number of session on the
target
max_agents_per_task                     4           (optional)Max number of agents to
use in parallel for each table/database/journal being moved
overwrite_existing_objects              true        (optional)Overwrites objects that
already exist on the target
force_utility                           dsa         (optional)Force DM to use a specific
utility for all DM operations
log_level                               2           (optional)Log level to output a log
file
online_archive                          true        (optional)Allows read/write access to the
source tables while the tables are being copied
table                                  DB1.TABLE   (optional)Table to be copied
response_timeout                        60         (optional)Amount of time to wait
for Daemon response in seconds
uowid                                  uowid       (optional)Unit of work id to identify
the job execution.
security_username                       dmcl_admin  (optional)User ID of the super user.
Only used if security management is enabled.
security_password                       53cUr17y   (optional>Password for the super
user. Only used if security management is enabled.
security_password_encrypted              052c7aab1.. (optional)Encrypted password for
```

```

the super user. Only used if security management is enabled.
query_band          AppName=B;      (optional)QueryBand passed to the
Database as a list of name=value pairs in a string.
netrace             21              (optional)cli netrace parameter
value.
netrace_buf_len     0               (optional)cli netrace_buf_len
parameter value.
tpt_debug           1              (optional)tptapi trace debug parameter
value.
source_staging_database db1        (optional)source staging database
name
target_staging_database db1        (optional)target staging database
name
target_database     db1            (optional)target database name
db_client_encryption true          (optional)set if job need to encrypted
during data transfer.

```

Using the Command-Line Interface

After the command-line interface is installed, you can run a Data Mover command from any directory by typing `datamove` followed by a command and any required parameters.

Commands generally have required and optional parameters that can be specified by typing them directly in the command line, after the entered command. Many of the commands require or allow you to pass additional information in an XML file. In addition, you can refer to sample XML files that are included when you install the command-line interface package. The sample XML files are located at `/opt/teradata/client/nn.nn/datamover/commandline/samples`, where *nn.nn* refers to the major and minor version number of the product (for example, 16.20).

Command-Line Interface Exit Codes

The command-line interface returns exit codes indicating results of running commands.

Exit Code	Description
0	Command ran successfully
Other than 0	An error occurred

Note:

Ensure error codes in automated scripts generate 0 when commands run successfully and a value other than 0 when an error occurs.

Data Mover Command Types

Data Mover commands can be categorized as two major types:

Command	Description
Setup Commands	Enables you to set up and configure the Data Mover system, including the daemon and agents. This functionality corresponds to that provided by the Data Mover Setup portlet.
Job Management Commands	Enables you to create, run, monitor, update, and delete jobs. This functionality corresponds to that provided by the Data Mover portlet.

The following table describes the administration and configuration Data Mover commands.

Command Name	Description
backup_daemon	Performs a backup of the Data Mover and Cloud Staging Copy Service repository.
create_cloud_staging	Create a cloud staging area to be used for a Data Mover cloud staging copy jobs.
create_event_table	Creates an event table for logging Teradata Ecosystem Manager events.
delete_cloud_staging	Deletes the specified cloud staging area.
delete_event_table	Deletes an event table.
edit_cloud_staging	Edits the specified cloud staging area.
get_cloud_staging	Gets the details of the specified cloud staging area.
help	Displays a brief description of all of the commands, and allows you to view further details about each, including its parameters.
list_agents	Lists the Data Mover agents connected to the Data Mover daemon.
list_cloud_staging	Lists the existing cloud staging areas.
list_configuration	Outputs configuration and performance settings for the Data Mover daemon.
list_event_table	Lists all existing event tables.
modify_admin_password	Changes the password for the super user. By default, the password for the super user is dmc1_admin. The user ID of the super user is dmc1_admin and cannot be changed.
modify_event_table	Changes the user name or user password for an existing event table.
restore_daemon	Restores previously backed up Data Mover daemon repository, Cloud Staging Copy Service repository, and properties files.
save_configuration	Saves the information in the configuration file that was created by the list_configuration command to the Data Mover daemon.

The following table describes the job management and reporting commands.

Command Name	Description
cleanup	Cleans up Teradata DSA, Teradata PT API, and Teradata JDBC tasks from a job that failed or was stopped before completing successfully.
create	Creates a copy job based on the syntax parameters and the object list.
delete_job	Deletes the specified job.
edit	Changes the definition for the specified job name.
encrypt_password	Creates an encrypted password from a password entered as a parameter on the command line, interactively, or through an XML file with a specific format.
list_job_definition	Lists the specified job definition.
list_jobs	Displays a list of all jobs that have run on the Data Mover daemon with the specified status mode. Job definitions are not displayed.
list_job_steps	Displays all of the steps in a job plan.
list_tasks	Lists the status of all active tasks from all jobs currently in the RUNNING state.
move	Creates a job to copy specified database objects from one database to another, then starts the job immediately.
restart	Restarts a job that has failed.
start	Starts a job that was created with the create command.
status	Displays the status of a specified job.
stop	Stops a running job.
update_job_priorities	Updates priority for one or more jobs in queue using only the parameters.xml file; not available directly in the command line.
update_job_steps	Refreshes the job steps of a specified job.

Properties File for the Command-Line Interface

The `commandline.properties` file configures the command-line interface. It is installed as part of the Data Mover command-line installation process.

Note:

The Data Mover REST server URL is specified in the first half of the output. If you want to connect to a different REST server (and thus a different daemon) at runtime, you can change these default values by entering them as parameters of the Data Mover commands.

```
# Copyright (C) 2009-2021 by Teradata Corporation.
# All Rights Reserved.
# TERADATA CORPORATION CONFIDENTIAL AND TRADE SECRET
```

```

#-----
-----
# File: "commandline.properties"
#
# Purpose: This file contains all of the properties used by the DM CommandLine.
#
# Caution: Do not modify the property names/values in this file unless absolutely
sure
# of the implications to the DM CommandLine.
#
# Note: Any line of text in this document that begins with a '#' character is a
comment and
# has no affect on the DM CommandLine. However, comments should not be modified.
#
# All properties under LOGGING Comment are used for logging purposes
#
#-----
-----
# Purpose: Data Mover Rest Interface
# Default: https://localhost:1443/datamover
# For automatic failover support:
# dm.rest.endpoint=https://activeServer:1443/datamover,https://
standbyServer:1443/datamover
dm.rest.endpoint=https://localhost:1443/datamover

#-----LOGGING-----

# Purpose: Set Logging level to info.  User has 6 options.
# From most amount of logging to least: trace < debug < info < warn < error <
fatal
rootLogger.level=info

# Purpose: Informs the logging application to use a specific appender and it's
properties.  DO NOT CHANGE
appender.rolling.type=RollingFile
appender.rolling.name=RollingFile
appender.rolling.layout.type=PatternLayout
appender.rolling.layout.pattern=%d [%t] %-5p %c{3}(%L) - %m%n
appender.rolling.policies.type=Policies
appender.rolling.policies.size.type=SizeBasedTriggeringPolicy
appender.rolling.strategy.type=DefaultRolloverStrategy
logger.rolling.name=com.teradata
logger.rolling.appenderRef.rolling.ref=RollingFile

```



```
# Purpose: Allow's user the ability to change the location of the log file.
# If changing log file location, please give the absolute path of the file;
# for example, /var/log/dmCommandLine.log
# for windows os: use slash instead of back slash:
# For Example: C:/Program File/Teradata/Log/dmCommandLine.log
appender.rolling.fileName=dmCommandLine.log
appender.rolling.filePattern=dmCommandLine.log.%i

# Purpose: The max size of the logging file before being rolled over to backup
files.
appender.rolling.policies.size.size=20MB
# Purpose: The number of backup logging files created, after the max number, the
oldest file is erased.
appender.rolling.strategy.max=5

#-----
```

Data Mover XML Files

Data Mover commands have required and optional parameters you can specify by typing them directly in the command line or in an XML file the command uses. It is often more convenient to use XML files because you can reuse and easily modify them without having to type the parameter values each time you run a command.

The XML files that specify command parameters are referred to as `parameters.xml` throughout this book.

If the same parameter is defined in `parameters.xml` and also directly in the command-line, a message displays and the value from the command-line takes precedence.

Case Sensitivity in XML Files

When you create or modify an XML file, you specify values for elements, which correspond to command parameters. Element names, which are enclosed by opening and closing XML tags, are case-sensitive and must appear as shown in the sample XML files. For example, the `job_name` element must be in lowercase letters and must not be changed to `JOB_NAME`.

Some of the values that you specify or change are not case-sensitive. For example, the value for the element for `export_without_spool` can be entered as either `true` or `TRUE`. Both are valid. For the `create` command, the following two example XML files are equivalent.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!-- lowercase values -->
<dmCreate xmlns="http://schemas.teradata.com/dataMover/v2009"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://schemas.teradata.com/unity/datamover.xsd">
  <job_name>floyd_dmdev_create</job_name>
  <source_tdpid>floyd</source_tdpid>
  <source_user>dmquest</source_user>
```

```

<source_password>please</source_password>
<target_tdpid>dmdev</target_tdpid>
<target_user>dmguest</target_user>
<target_password>please</target_password>
<data_streams>4</data_streams>
<max_agents_per_task>4</max_agents_per_task>
<source_sessions>4</source_sessions>
<target_sessions>4</target_sessions>
<force_utility>tptapi</force_utility>
<log_level>0</log_level>
    <database selection="unselected">
        <name>dmguest</name>
        <table selection="included">
            <name>orders_2010</name>
            <export_without_spool>true</export_without_spool>
            <validate_row_count>partial</validate_row_count>
            <sql_where_clause>
                <![CDATA[WHERE colA > 100]]>
            </sql_where_clause>
            <key_columns>
                <key_column>colA</key_column>
            </key_columns>
            <staging_to_target>insert_only</staging_to_target>
        </table>
    </database>
</dmCreate>

<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!-- UPPERCASE VALUES -->
<dmCreate xmlns="http://schemas.teradata.com/dataMover/v2009"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://schemas.teradata.com/unity/datamover.xsd">
    <job_name>floyd_dmdev_create</job_name>
    <source_tdpid>floyd</source_tdpid>
    <source_user>dmguest</source_user>
    <source_password>please</source_password>
    <target_tdpid>dmdev</target_tdpid>
    <target_user>dmguest</target_user>
    <target_password>please</target_password>
    <data_streams>4</data_streams>
    <max_agents_per_task>4</max_agents_per_task>
    <source_sessions>4</source_sessions>
    <target_sessions>4</target_sessions>
    <force_utility>TPTAPI</force_utility>
    <log_level>0</log_level>
    <database selection="unselected">
        <name>dmguest</name>
        <table selection="included">
            <name>orders_2010</name>
            <export_without_spool>TRUE</export_without_spool>
            <validate_row_count>PARTIAL</validate_row_count>
            <sql_where_clause>
                <![CDATA[WHERE colA > 100]]>
            </sql_where_clause>
            <key_columns>
                <key_column>colA</key_column>
            </key_columns>
            <staging_to_target>INSERT_ONLY</staging_to_target>
        </table>
    </database>
</dmCreate>

```

The following table lists values that are not case-sensitive:

Element	Value
force_utility	DSA
	TPTAPI
	TPTAPI_LOAD
	TPTAPI_UPDATE
	TPTAPI_STREAM
	JDBC
	T2T
export_without_spool overwrite_existing_objects online_archive	TRUE
	FALSE
	FALSE
	UNSPECIFIED
force_target_staging_table	TRUE
	FALSE
validate_row_count	NONE
	PARTIAL
	ALL
staging_to_target	NOT_SPECIFIED
	DELETE_INSERT
	MERGE
	INSERT_ONLY
status_mode	A
	N
	I
	R
	C
	F
	RS
	Q
	UC

Element	Value
	B
action_time	BEFORE
	AFTER
enabled (an attribute of the action_time element)	YES
	NO
index_type	HASH_INDEX
	JOIN_INDEX

The values TRUE and FALSE are not case-sensitive for several elements and attributes, including:

- compare_ddl
- override_lock_access
- use_userid_pool
- sync (a child element of the dmCreate and dmEdit elements)
- all and skip_prompt (child elements of the delete_job element)
- copyStats (an attribute of the table or index elements)

The Data Mover XML Schema

The Data Mover XML Schema describes messages that Data Mover components use to communicate over the JMS bus. A copy of the schema is included in the command-line utility directory, and is also available at [Data Mover XML Schemas](#).

The Data Mover XML Schema describes the structure of the XML files that can be used by the Data Mover command-line interface. The schema can be used for validation if a compatible XML authoring tool is used.

Data Mover supports XML parameter files in ASCII and UTF-8. The command-line Console supports only ASCII.

The code sample shows part of the dmCreate element of the Data Mover XML schema.

Note:

The order of elements shown in the Data Mover XML schema must be followed.

```
<xsd:element name="dmCreate">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="jobDefinitionType">
        <xsd:sequence>
          <!-- permission setting for the created job -->
          <xsd:element name="job_security" minOccurs="0" type="securityType"/>
          <xsd:element name="command_start_time" minOccurs="0"
type="xsd:dateTime"/>
        
```

```

    </xsd:sequence>
  </xsd:extension>
</xsd:complexContent>
</xsd:complexType>
</xsd:element>

```

The following code sample shows a valid Data Mover message that has been produced by using the message format described in the Data Mover XML Schema.

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<dmCreate xmlns="http://schemas.teradata.com/dataMover/v2009"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://
schemas.teradata.com/unity/DataMover.xsd">
  <job_name>NAME</job_name>
  <job_priority>UNSPECIFIED</job_priority>
  <source_tdpid>SourceIP</source_tdpid>
  <source_user>USER</source_user>
  <source_password>PASS</source_password>
  <target_tdpid>TargetIP</target_tdpid>
  <target_user>USER</target_user>
  <target_password>PASS</target_password>
  <database selection="unselected">
    <name>DatabaseName</name>
    <table selection="included">
      <name>TableName</name>
    </table>
  </database>
</dmCreate>

```

Schema Versions

The Data Mover XML Schema uses the same namespace for different versions of the schema. Each schema file contains a version number that you must declare in the root element of your XML document, as shown in the example here:

```

<?xml version="1.0" encoding="UTF-8" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://schemas.teradata.com/dataMover/v2009"
  xmlns="http://schemas.teradata.com/dataMover/v2009"
  elementFormDefault="qualified" attributeFormDefault="unqualified">

```

The following example shows the root element of an XML document that contains the version:

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<dmCreate xmlns="http://schemas.teradata.com/dataMover/v2009"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://
schemas.teradata.com/unity/DataMover.xsd">

```

Sample Job Creation XML File

The following sample XML illustrates the correct order of all possible elements available for a job creation XML file.

Refer to this sample XML before creating a job.

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<dmCreate xmlns="http://schemas.teradata.com/dataMover/v2009" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://
schemas.teradata.com/unity/datamover.xsd">
  <job_name>datamovertest</job_name>
  <source_tdpid>dmdev</source_tdpid>
  <source_user>dmguest</source_user>
  <source_password>please</source_password>
  <source_password_encrypted></source_password_encrypted>
  <source_logon_mechanism></source_logon_mechanism>
  <source_logon_mechanism_data></source_logon_mechanism_data>
  <source_account_id></source_account_id>
  <source_session_charset></source_session_charset>
  <target_tdpid>dmsmp2</target_tdpid>
  <target_user>dmguest</target_user>
  <target_password>please</target_password>
  <target_password_encrypted></target_password_encrypted>
  <target_logon_mechanism></target_logon_mechanism>
  <target_logon_mechanism_data></target_logon_mechanism_data>
  <target_account_id></target_account_id>
  <target_session_charset></target_session_charset>
  <use_userid_pool>>false</use_userid_pool>
  <data_streams>1</data_streams>
  <source_sessions>1</source_sessions>
  <target_sessions>1</target_sessions>
  <max_agents_per_task>1</max_agents_per_task>
  <response_timeout>100</response_timeout>
  <sync>>true</sync>
  <overwrite_existing_objects>>true</overwrite_existing_objects>
  <freeze_job_steps>>false</freeze_job_steps>
  <force_utility>tptapi</force_utility>
  <staging_database>
    <name>dmguest</name>
  </staging_database>
  <staging_database_for_table>
    <name>dmguest</name>
  </staging_database_for_table>
  <target_database>
    <name>dmguest</name>
  </target_database>
  <log_level>0</log_level>
  <online_archive>>false</online_archive>
  <log_to_event_table>tmsmevent</log_to_event_table>

```

```

<database selection="unselected" replaceDatabase="true">
  <name>dmguest</name>
  <staging_database>
    <name>dmguest</name>
  </staging_database>
  <staging_database_for_table>
    <name>dmguest</name>
  </staging_database_for_table>
  <target_database>
    <name>dmguest</name>
  </target_database>
  <compare_ddl>TRUE</compare_ddl>
  <table selection="included" copyStats="false">
    <name>ppiorders</name>
    <staging_database>
      <name>dmguest</name>
    </staging_database>
    <staging_database_for_table>
      <name>dmguest</name>
    </staging_database_for_table>
    <force_target_staging_table>false</force_target_staging_table>
    <target_database>
      <name>dmguest</name>
    </target_database>
    <target_name>table8_clone</target_name>
    <validate_row_count>all</validate_row_count>
    <override_lock_access>false</override_lock_access>
    <export_without_spool>true</export_without_spool>
    <compare_ddl>TRUE</compare_ddl>
    <sql_where_clause><![CDATA[ WHERE ORDERDATE > '2003-01-23' ]]></
sql_where_clause>
    <key_columns>
      <key_column>ORDERDATE</key_column>
    </key_columns>
    <staging_to_target>insert_only</staging_to_target>
  </table>
  <table selection="included" copyStats="false">
    <name>employee</name>
    <compare_ddl>true</compare_ddl>
  </table>
</database>
<triggers>
  <trigger selection="included">
    <database>dmguest</database>

```

```

    <subject_table_database>dmguest</subject_table_database>
    <table>employee</table>
    <name>TestTrigger</name>
    <action_time enabled="NO">BEFORE</action_time>
</trigger>
</triggers>
<indices>
    <index selection="included" copyStats="false">
        <name>Orders_HI</name>
        <target_name>orders_HI_new</target_name>
        <index_database>dmguest</index_database>
        <index_type>HASH_INDEX</index_type>
    </index>
</indices>
<views>
    <view selection="included" copyData="true">
        <name>testview</name>
        <database>dmguest</database>
        <view_data_table>
            <target_table>TargetTable</target_table>
            <target_database>dmguest</target_database>
        </view_data_table>
        <staging_database>
            <name>dmguest</name>
        </staging_database>
        <staging_database_for_table>
            <name>dmguest</name>
        </staging_database_for_table>
        <force_target_staging_table>false</force_target_staging_table>
        <validate_row_count>all</validate_row_count>
        <compare_ddl>true</compare_ddl>
        <sql_where_clause><![CDATA[ WHERE "test2"."C1" > 500 ]]></
sql_where_clause>
        <key_columns>
            <key_column>C1</key_column>
        </key_columns>
    </view>
</views>
<foreign_servers>
    <foreign_server selection="included" >
        <name>FSObj</name>
        <map>map1</map>
        <colocate>col1</colocate>
    </foreign_server>

```



```

</foreign_servers>
<function_aliases>
  <function_alias selection="included" >
    <name>dmguest</name>
    <database>faObj</database>
  </function_alias>
</function_aliases>
<macros>
  <macro selection="included">
    <name>macro_employee</name>
    <database>dmguest</database>
  </macro>
</macros>
<stored_procedures>
  <stored_procedure selection="included">
    <name>getEmpInfo</name>
    <database>dmguest</database>
  </stored_procedure>
</stored_procedures>
<uowid>test</uowid>
<query_band>AppName=B;</query_band>
</dmCreate>

```

Data Mover Setup and Configuration

The command-line interface provides commands for setting up and configuring the Data Mover system. You may use the command-line interface setup and configuration commands as an alternative to using the **Data Mover Setup** portlet.

Commands for managing the Data Mover daemon and agents:

```

list_configuration
list_agents
save_configuration
backup_daemon
restore_daemon
modify_admin_password

```

Commands for managing event tables that log Teradata Ecosystem Manager events:

```

create_event_table
list_event_table
modify_event_table
delete_event_table

```

Configuring the Daemon

The properties for daemon configuration are contained in the `configuration.xml` file. This file includes properties for the following configurations:

- Preventing certain systems from being copy targets
 - Detecting unresponsive and blocked jobs
 - Configuring security
 - Controlling the maximum load slots Data Mover can use on a target system
1. Run the `list_configuration` command.
If you do not specify a different output path and filename, the `list_configuration` command writes its output to `./configuration.xml`.
 2. Edit the property values in the resulting configuration file.
The number or text enclosed in the `value` element is the only area eligible for editing. The text in the `description` element provides additional information only. Do not change the text enclosed in the `key` element.
 3. Run `datamove save_configuration` to save and apply the updated configuration information.

Skip Fastload and Multiload Tables

The `skip.FLML.tables` daemon property manages the job output status if selected objects within the running job are being fastloaded or multiloaded by another process.

The `skip.FLML.tables` values are as follows:

Value	Description
True	The job completes with a <code>completed_with_warnings</code> when fastload or multiload objects exist in the database being copied.
False	(Default) Job fails when fastload or multiload objects exist in the database being copied.

The following is the XML property definition added to the daemon configuration:

```
<property>
  <key>skip.FLML.tables</key>
  <value>true</value>
  <description>
    If enabled (true) and the table is being Fast/Multi
    Loaded, the daemon marks it as COMPLETED WITH WARNINGS, for DSA jobs. If disabled
    (false) and the table is being Fast/Multi Loaded, the daemon marks the table
    movement as FAILED, for DSA jobs. Default is false.
  </description>
</property>
```

Designating Non-Target Systems

You can prevent one or more Teradata Database systems from being a target system for a Data Mover job. This option ensures data is not copied to an unintended target system when running a job.

The systems may be specified either by their TDPID hostnames or their IP addresses, but must be consistent with your system specification in any Data Mover commands. For example, if you specify systems by using their IP addresses for Data Mover commands, you must specify non-target systems by their IP addresses.

1. Run the `list_configuration` command to generate `configuration.xml`.
2. Open `configuration.xml` for editing.
3. For the key `job.never.target.system` under the `property` element, set the `value` element to `true`. The default value for this key is `false`.
4. Add the `systemList` element under `value`.
5. For each non-target system, add a `targetSystem` element and enter the hostname TDPID or IP address of the system.
6. Close the entry with the `/targetSystem` element.
7. Save `configuration.xml` and run the `save_configuration` command to update the daemon with the new list of non-target systems.

In the following example `configuration.xml`, the Teradata Database systems with TDPID hostnames `PROD2` and `PROD3` cannot be used as target systems.

```
<property>
  <key>job.never.target.system</key>
  <value>true</value>
  <systemList>
    <targetSystem>PROD2</targetSystem>
    <targetSystem>PROD3</targetSystem>
  </systemList>
</property>
```

Scenarios for Non-Target System Designation

The following table describes what happens in specific scenarios with the `job.never.target.system` option.

Values	Effect
The value of <code>job.never.target.system</code> is <code>false</code> , and no systems are defined under <code><systemList></code>	Data Mover does not use this feature. This is the default setting.
The value of <code>job.never.target.system</code> is <code>false</code> , and non-target systems are defined under <code><systemList></code>	Data Mover ignores the list of non-target systems.

Values	Effect
The value of <code>job.never.target.system</code> is <code>true</code> , but no systems are defined under <code><systemList></code>	Data Mover returns an error.
The value of <code>job.never.target.system</code> is <code>true</code> , a non-target system is defined under <code><systemList></code> , and you attempt to create a Data Mover job that copies data to that system	Data Mover returns an error.
The non-target system defined under <code><systemList></code> is specified as an IP address, but the Data Mover command uses TDPID hostnames	Data Mover ignores the IP address in the list of non-target systems because Data Mover commands are using TDPID hostnames. You must use the same type of system specification for Data Mover commands and for this list.

Deadlock Retry

When Data Mover runs SQL on source and target systems, the SQL execution may fail if there is a deadlock on an object. This situation may cause the following Teradata Database error:

```
[Error 2631] [SQLState 40001] Transaction ABORTed due to Deadlock.
```

The 2631 error is a retryable error that may not occur if the SQL is started after an interval. There are three configuration properties in `configuration.xml` that you can configure to retry running the query:

Property	Description
<code>deadlock.retry.enabled</code>	If a SQL query execution fails with Teradata Database error 2631 due to a deadlock, retries running the query after a specified time interval. Default is <code>false</code> . Specify <code>true</code> to enable. The two properties listed here are disregarded if <code>deadlock.retry.enabled</code> property is set to <code>false</code> .
<code>deadlock.retry.interval</code>	An interval to retry running a SQL query that fails due to Teradata Database deadlock error 2631. Data Mover waits this interval before trying to run a failed query. Specify values for each of the two elements. The default is 1 minute. <ul style="list-style-type: none"> <code>value</code>: specify the number of seconds or minutes for the retry interval. Must be a positive integer. <code>unit</code>: specify SECONDS or MINUTES
<code>deadlock.retry.maxAttempts</code>	The maximum number of attempts to retry running a SQL query that fails with Teradata Database deadlock error 2631. If the SQL query fails after being retried for the number of times specified by this property, an error is returned to the user. The default is 10. Must be a positive integer.

The deadlock retry properties do not apply to queries run by the JDBC, Teradata PT API, or DSA operators during the extract and load process. The properties only apply to SQL run by Data Mover during job creation time or before or after data load time.

Detecting Unresponsive Jobs

Data Mover can be configured to detect and stop jobs that become unresponsive after user-defined timeout periods. Configure this feature by using the **Job Timeout** tab of the **Data Mover Setup** portlet or in the configuration.xml file generated when the list_configuration command is run.

Variable Timeout Periods

A Data Mover job includes several different phases, and the duration of some phases depends heavily on the size of the data being copied. The larger the size of the data copied, the longer the time required for the initiate phase, the apply rows phase (for Teradata PT API tasks), or the build phase (for DSA tasks). You can use the **Data Mover Setup** portlet or configuration.xml to specify the size category and standard timeout periods for these variable timeouts for different size categories and phases.

Data Mover performs the following process to determine the actual timeout period to use before stopping an unresponsive job:

1. Detects the size of the object a job is moving to determine whether the object is small, medium, or large.
2. Checks the current phase of the job and uses the timeout period for the phase to determine whether the job is taking longer than the timeout period.

Configuration Properties for Unresponsive Jobs

You can set the configuration properties listed in the following table to enable detection of unresponsive jobs and set the timeout period when these jobs are stopped.

Property	Description	Default Value
hanging.job.check.enabled	Enable or disable detection of unresponsive jobs. Note: If this property (hanging.job.check.enabled) is not set to true, none of the defaults for the other properties listed in this table are applicable.	false
hanging.job.check.rate	Frequency with which Data Mover checks for unresponsive jobs (in hours).	1
hanging.job.timeout.acquisition	Timeout for the acquisition phase of tasks (in hours).	1
hanging.job.timeout.range.small.max	Defines maximum size (in MB) for an object to be considered a small object.	5
hanging.job.timeout.range.large.min	Defines minimum size (in GB) for an object to be considered a large object.	10

Property	Description	Default Value
hanging.job.timeout.small.apply	Small objects: Timeout period for application phase (in hours)	2
hanging.job.timeout.small.build	Small objects: Timeout period for build phase (in hours)	2
hanging.job.timeout.small.initiate	Small objects: Timeout period for initiate phase (in hours)	2
hanging.job.timeout.medium.apply	Medium objects: Timeout period for application phase (in hours)	4
hanging.job.timeout.medium.build	Medium objects: Timeout period for build phase (in hours)	4
hanging.job.timeout.medium.initiate	Medium objects: Timeout period for initiate phase (in hours)	4
hanging.job.timeout.large.apply	Large objects: Timeout period for application phase (in hours)	8
hanging.job.timeout.large.build	Large objects: Timeout period for build phase (in hours)	8
hanging.job.timeout.large.initiate	Large objects: Timeout period for initiate phase (in hours)	8

Blocked Job Detection

Data Mover jobs may fail to complete successfully if they are blocked because there are locks on source or target database objects.

Data Mover provides four configuration properties that can be set to detect whether object locks exist, and to retry starting blocked jobs at a specified interval. These blocked job configuration properties can be edited in the `configuration.xml` file.

Note:

If blocked job detection is enabled, Data Mover must run a query for each table being copied. Therefore, there may be a noticeable decrease in performance when this feature is enabled. If you are moving an entire database, the queries are run one time on a randomly chosen table in the database to check for database locks.

Blocked Job Configuration Properties

Property	Description and Values
<code>blocked.job.retry.enabled</code>	<p>The property that determines whether Data Mover checks whether there are any locks on source or target objects prior to executing a job. If enabled and locks are detected, it tries again to run the job after a specified interval. If this property is not enabled, the other related properties listed here are ignored.</p> <p>Specify one of the following options for the value element:</p> <ul style="list-style-type: none"> • <code>true</code>: Data Mover detects locks and tries to start the job after the interval specified. The job is only run when there are no locks on objects. • <code>false</code>: Default. Data Mover does not check for object locks. If locks exist, the job hangs until locks are released.
<code>blocked.job.retry.interval</code>	<p>The time interval to wait if locks are detected before checking again whether the locks are still there. The default setting is 1 hour.</p> <p>Specify values for each of the two elements:</p> <ul style="list-style-type: none"> • <code>value</code>: specify the number of minutes or hours for the retry interval. Must be a positive integer. • <code>unit</code>: specify HOURS or MINUTES
<code>blocked.job.retry.maxInterval</code>	<p>The maximum interval for attempting to start any jobs blocked due to locks on source or target objects. Jobs are marked as failed after this interval is exceeded and they are still blocked. The default setting is 1 hour.</p> <p>Specify values for each of the two elements:</p> <ul style="list-style-type: none"> • <code>value</code>: specify the number of minutes or hours for the retry interval. Must be a positive integer. • <code>unit</code>: specify HOURS or MINUTES
<code>blocked.job.maxAllowedLimit</code>	<p>The maximum number of jobs that can be marked as blocked and re-tried. If a job is detected as blocked when this number is reached, the job is added to the job queue. The default value is 5. You may specify a different number for the value element, but the value cannot be more than 25% of the maximum concurrent job limit.</p>

Rules for the `blocked.job.maxAllowedLimit` property

- The `blocked.job.maxAllowedLimit` value cannot exceed 25% of the maximum concurrent job limit. This is designed to prevent blocked jobs from consuming too many slots and preventing other jobs from running.
- If a blocked job is detected when this limit has already been reached, the job is added to the job queue, even if the number of jobs already in the queue has reached the maximum queued jobs limit.
- Attempting to save `configuration.xml` with a value greater than 25% of the maximum concurrent job limit results in an error.
- If the maximum concurrent job limit value is changed when restarting the daemon, a value of 25% of the new maximum concurrent job limit is automatically used for the `blocked.job.maxAllowedLimit` property.

Starting a Blocked Job

The **sync** parameter of the Start command waits for a job to complete and then returns an exit code to indicate whether the job completed successfully. The effect of the **sync** parameter setting when trying to start a blocked job is outlined in the following table.

Sync parameter	Effect
Enabled	The command line waits until the job starts successfully or fails because the maximum retry interval is reached. The job is marked as failed if the locks are not released when the maximum interval is reached or if a daemon failure occurs while it is in a blocked state. Jobs that fail this way can be started again after the locks are released or when the daemon is restored.
Not enabled	The command line exits with a message that an attempt to start the job occurs when the locks are released. A background process attempts to check locks after the interval specified by <code>blocked.job.retry.interval</code> and starts the job if no locks are detected. During the time that a job waits for locks to be released, the job is in a blocked state, which is displayed in the Status output. A user can cancel a blocked status job at any time by running the stop command. A job in a blocked state consumes a slot from the maximum concurrent job limit until it runs or fails due to reaching the maximum retry interval.

Configuring Blocked Job Retry

You can set configuration properties to detect locks on source or target objects that prevent a job from running successfully. You can configure Data Mover to try to start a blocked job after the interval you specify.

1. Run the `list_configuration` command to generate `configuration.xml`.
2. Open the `configuration.xml` file.
3. For the `blocked.job.retry.enabled` key, specify the one of the following options:

Option	Description
true	Enables detection of locks on the source and target objects included in a job.
false	Default value. Data Mover does not check to see whether there are locks on source or target objects. If there are locks, the job stops.

4. For the `blocked.job.retry.interval` key, specify the time interval for Data Mover to wait before rechecking whether objects are still locked:

Option	Description
value	Enter the number of minutes or hours.
unit	Enter the time unit. Valid entries are: MINUTES , HOURS .

- For the `blocked.job.retry.maxInterval` key, specify the maximum amount of time to wait when checking object locks before exiting, indicating an error message and marking the job as failed.

Option	Description
value	Enter the number of minutes or hours.
unit	Enter the time unit. Valid entries are: MINUTES , HOURS .

Make sure the value is always larger than the value specified for the `blocked.job.retry.interval` key.

- For the `blocked.job.maxAllowedLimit` key, specify the maximum number of jobs that can be marked as blocked and retried.
- Save `configuration.xml` and run the `save_configuration` command to apply the blocked job configuration settings.

Managing Credential Pools

Create credential pools for source and target systems in the `configuration.xml` file to allow job changes at a global level. Each pool has a unique name, defines credentials for both source and target systems, and defines one or more users for each system. When creating a job, after specifying the credential pool by name, Data Mover examines the pool for the source and target system names and user profiles to randomly pick an available user for that job. If multiple DSA jobs are running that have the same target system, an available user from the pool that is not already being used by other DSA jobs is selected.

In addition to being able to pick from a pool of users for the specified systems, credential pools enable you to change passwords in one place for all users specified in the pool.

The `job.useGroupUserIdPool` property is included in the `configuration.xml` file as shown here:

```
<property>
  <key>job.useGroupUserIdPool</key>
  <value>>false</value>
  <description>Purpose: Use a source or target user from the pool of
  users. This enables changing password in a single place</description>
</property>
```

In the following `configuration.xml` example, a `job.useGroupUserIdPool` is true and two credential pools have been created.

```
<property>
  <key>job.useGroupUserIdPool</key>
  <value>true</value>
  <groupPools>
    <groupPool>
      <name>POOL-1</name>
      <system name="system1">
```

```

        <user>
            <name>admin</name>
            <password>admin</password>
            <encrypted_password></encrypted_password>
        </user>
        <!-- more users -->
    </system>
    <!-- more systems -->
</groupPool>
<groupPool>
    <name>POOL-2</name>
    <system name="system2">
        <user>
            <name>dbc</name>
            <password>dbc</password>
            <encrypted_password></encrypted_password>
        </user>
        <!-- more users -->
    </system>
    <system name="system3">
        <user>
            <name>user1</name>
            <password>pass1</password>
            <encrypted_password></encrypted_password>
        </user>
        <!-- more users -->
    </system>
    <!-- more systems -->
</groupPool>
</groupPools>
    <description>Purpose: Use a source or target user from the pool of users.
    This enables changing password in a single place.</description>
</property>

```

In the following example of a job definition, a group user pool is used.

```

<source_tdpid>system1</source_tdpid>
<source_user></source_user>
<source_password></source_password>
<source_userid_pool>POOL-1</source_userid_pool>

<target_tdpid>system2</target_tdpid>
<target_user></target_user>

```

```
<target_password></target_password>
<target_userid_pool>POOL-1</target_userid_pool>
```

Usage Notes

- The name reservedUserPool cannot be used when defining a credential pool.
- Trying to specify source_userid_pool or target_userid_pool in the job definition when job.useGroupUserIdPool is set to false in configuration.xml results in an error.
- Trying to provide source_user and source_userid_pool in the same job definition results in a create time error. Only one method may be used for the source system. Data Mover chooses the user for the source system automatically from the specified pool when source_userid_pool is used.
- Trying to provide target_user and target_userid_pool in the same job definition results in a create time error. Only one method may be used for the target system. Data Mover chooses the user for the target system automatically from the specified pool when target_userid_pool is used.
- When source_userid_pool is specified and use_userid_pool is true or when target_userid_pool is specified and use_userid_pool in the same job definition, a create time error occurs.
- Different credential methods may be used for source and target. Providing source_user and target_userid_pool in the same job definition is valid. Providing target_user and source_userid_pool in the same job definition is also valid.
- When providing both source_userid_pool and target_userid_pool in the same job definition, the same credential pool must be used for both. Trying to specify one credential pool for the source and a different one for the target results in a create time error.
- When target_userid_pool is set, Data Mover chooses a user from the pool. If multiple DSA jobs are running at the same time and target_userid_pool is set and the pool has multiple users for the target system, Data Mover chooses an available user from the pool that is not already being used by other DSA jobs. This allows multiple DSA jobs to run at the same time.
 - For Teradata PT API/JDBC jobs, Data Mover may choose the same target user for two or more jobs being run at the same time since there is no problem in multiple users being logged on to a target system at the same time.

For DSA job source users, there are no restrictions to have the same user access the same system; therefore, Data Mover can choose the same user as the source user for different DSA jobs.

- When running a DSA job with target_userid_pool set and all target users in the pool are being used by other DSA tasks, the DSA task waits for the other tasks to complete and return the user to the pool. When the user is available in the pool, the waiting task picks up the user.
- All users in the group user pool are marked as available every time the daemon is restarted or the job.useGroupUserIdPool configuration property is modified. Once a user is locked in one group, the same user for the same system in different groups (including in target user ID pool) is locked. Similarly, once a user is unlocked, the same user for the same system in different groups (including in target user ID pool) is unlocked.

- If a DSA job does not use the pool but defines a target system and target user that have been assigned to a pool, that user is marked unavailable by the pool until the job is finished.
- If the target user specified in the job definition is part of a pool and that user is being used by another DSA job that uses the pool, the job fails because DSA does not allow the same user to logged on to the same target system at the same time.
- The credential pools feature is only available on Teradata systems.

Creating Credential Pools

The `configuration.xml` file is used to create one or more systems and users associated with one or more credential pools.

1. Run `datamove list_configuration` to generate `configuration.xml`.
2. Edit `configuration.xml` in the section under `<key>job.useGroupUserIDPool</key>` and enter the following values:

Parameter	Value
value	true or false True allows Data Mover to select any available source or target systems or users specified in the pool. False (default) does not select any source and target systems or users specified in the pool for Data Mover jobs.
system.name	Enter the name of the system.
name	Enter the name of the user.
password	Add a value to use an unencrypted password.
encrypted_password	Enter a value to use an encrypted password.

3. Once a credential pool is created, and the property `job.useUserIdPool` is set to true in `configuration.xml`, Data Mover selects an available source or target system from the specified pool when running jobs.

Data Mover Security

Data Mover provides configuration parameters that control how security permissions are used. When security management is enabled, the access privileges available to a user for the daemon and for individual jobs depend on the security settings designated. If security management is not enabled, a Viewpoint user can perform any operation on any job in the **Data Mover** portlet.

Security Configuration Parameters

You can set the following security configuration parameters in the `configuration.xml` file you generate using the `list_configuration` command.

Parameter	Description
<code>job.useSecurityMgmt</code>	Determines whether security management is used. When set to <code>true</code> , the security framework is enabled, and the following two security parameters apply. The default is <code>true</code> .
<code>job.securityMgmtLevel</code>	Determines the level of security management. The valid choices are <code>daemon</code> and <code>job</code> . The default is <code>job</code> .
<code>job.allowCommandLineUser</code>	Determines whether the daemon always allows command line requests when the security level is set to <code>daemon</code> . When set to <code>true</code> , the command line does not enforce security checking even if security is enabled for the portlet. The default is <code>false</code> .

User Profiles

When a user logs onto the **Data Mover** portlet, a user profile is authorized. The user profile contains one user name and a list of roles to which the user belongs. The user profile determines what actions the user can do, depending on whether global or job level permissions apply.

Daemon-Level Permissions

When the `job.useSecurityMgmt` parameter is set to `daemon`, daemon-level permissions are used. A user profile is checked for its daemon read, execute, and write permissions. If the user name or any role of a user profile has a permission (read, run, or write), the user profile has the permission. The permission applies to all jobs on the daemon.

Job-Level Permissions

When the `job.useSecurityMgmt` parameter is set to `job`, both the daemon and the job level permissions are evaluated. A user profile has permission on a specific job only if the user profile has that daemon permission and the job level permission on that job. If the user name or any role of a user profile has a job-level permission (read, run, or write), permission is granted to the user profile for that particular job.

Read, write, and run permissions are assessed independently of each other. For example, a user or role has execute permissions for a job only if that user or role has execution permission at both the daemon level and at the job level. The same applies to read and write permissions. If a user profile contains multiple roles, the user profile is granted permissions if one role has daemon permissions and another has job level permissions.

Command-Line Use by Viewpoint Users

When security management is enabled, Viewpoint users can be authenticated to use the Data Mover command line interface. The Viewpoint host name and port must be configured in the `daemon.properties` file, as shown in this snippet:

```
# Purpose: The hostname or IP address for the ViewPoint Authentication server.
# Default: https://localhost
viewpoint.url=https://localhost
```

```
# Purpose: The port number for the ViewPoint Authentication server.
# Default: 443
viewpoint.port=443
```

Note:

The Viewpoint server used for authentication must be the only Viewpoint server monitoring this daemon. Monitoring the same daemon on more than one Viewpoint server is not supported and could create authentication and job permission issues. This restriction applies regardless of whether security management is currently enabled on the daemon or not.

If the Viewpoint Authentication server does not have HTTPS enabled, you can set the following if you want to authenticate using HTTP instead: `viewpoint.url` to `http://localhost` and `viewpoint.port` to 80.

The Data Mover daemon makes the web services call to authenticate the user. The HTTP based service call URL is in this format: `http://hostname: port /ws/security/rolesForCurrentUser`.

Enable or Disable Data Mover Security Management

You can enable or disable Data Mover to restrict the running of certain commands for the command-line interface and job actions in the Data Mover portlet by editing the `configuration.xml` file for the daemon.

1. Run the `list_configuration` command.
2. Open the resulting configuration file.
3. For the key `job.useSecurityMgmt`, set the value element to `true` or `false`.
The default value for this key is `true`.
4. For the key `job.securityMgmtLevel`, set the value element to either `daemon` or `job`.
5. For the key `job.allowCommandLineUser`, set the value element to either `true` or `false`.
6. Run the `save_configuration` command to enable security management on the daemon.

Access Permissions for Data Mover Commands

When security is not enabled, a command line user has permission to run all Data Mover commands. When security management is enabled and daemon-level security is selected, the rules described here apply to the command line interface.

If the security level is set at daemon level, only daemon-level permission is checked. If the security level is set to job, both daemon- and job-level permissions are evaluated.

A user needs job-level (if security is set to job level) and daemon-level read permission to run the following commands:

- `list_jobs`
- `list_job_definition`
- `list_job_steps`

- status

A user needs job-level (if security is set to job level) and daemon-level write permission to run the following commands:

- delete_job
- move
- create
- edit
- update_job_steps
- update_job_priorities

A user need job-level (if security level is job) and daemon-level permission to run the following commands:

- start
- stop
- restart
- cleanup
- move

A user needs to be either a command-line interface super user (dmcl_admin) or Viewpoint Administrator user to run these Admin-level commands:

- backup_daemon
- restore_daemon
- list_event_table
- create_event_table
- delete_event_table
- modify_event_table
- list_configuration
- save_configuration
- modify_admin_password

Security Settings Rules

When the security management framework is enabled, the following rules apply:

- The super user (dmcl_admin) can run any command.
- A user who has write or execute permission implicitly has read permission also. This applies to both daemon and job-level permissions.
- A user who creates a job is the job owner and automatically has job-level read, write, and execution permissions for the job.

- The job-level read, write, and execute permissions are applied at the base job name only. For commands that have the job execution name in the **job_name** parameter, the job-level permission is checked against the execution base job name.
- When a regular Viewpoint user runs the create command to create a job, the owner_name field in the SecurityType object is replaced in the create job request to represent the user authentication result. This occurs whenever the request user name is not the super user (dmcl_admin), regardless of the daemon security setting. The Data Mover daemon processes the create command and records the owner name and other user and role permission information. For the super user (dmcl_admin), the owner_name is not changed. This allows the super user to run the create job command with any user credential provided in the original job request.
- When daemon security is enabled, user global modification permissions are verified when running the create command. This includes properties such as available utilities and maximum number of streams. If a user does not have the proper modification permission, the create request fails.
- When daemon security is enabled, user global modification permissions are verified for update_job_steps and update_job_priorities. If a user does not have the proper modification permission, a security exception occurs.
- A user can update job permissions by running the start command with dynamical parameters, or by using the edit_job command. When daemon security is enabled, a user must be the super user (dmcl_admin), the Viewpoint Administrator with write permissions, or the job owner to update job permissions; otherwise, a security exception occurs.
- When security is enabled, only the super user (dmcl_admin) or Viewpoint Administrator with write permission can use the start or edit job command to change job owner.

Parameters for Commands When Security Management is Enabled

When security management is enabled, Viewpoint users and Data Mover super-users must specify values for several security parameters.

Viewpoint Users

A Viewpoint user must provide their Viewpoint user name and password using the **security_username** and **security_password** parameters. When a user provides a value for the **security_username** parameter, the Viewpoint authentication process returns information about the user's role to the command line. If the user's role has the required access permission to run the command, the command is executed.

The Viewpoint user can provide the security parameters directly on the command line, or in a parameters.xml file. For example, a Viewpoint user who wants to run the list_jobs command could type:

```
datamove list_jobs -security_username ViewpointUser -security_password
MyPassword -job_name MyJob
```

Alternatively, the user could type:


```
datamove list_jobs -f list_job.xml
```

where list_job.xml contains the following:

```
<dmListJobs xmlns="http://schemas.teradata.com/dataMover/v2009"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schemas.teradata.com/unity/datamover.xsd">
  <dmSecurity>
    <security_username>user</security_username>
    <security_password>test</security_username>
  </dmSecurity>
  <job_name>test</job_name>
</dmListJobs>
```

Regardless of which method used, if **security_password** is not specified or is left empty, the command line interface will prompt for the Viewpoint password. The user can type in the masked Viewpoint password at that time.

Data Mover Super User

Certain commands can only be run by Data Mover super users or Viewpoint Administrators. Data Mover super users must specify values for one of the following either directly on the command-line interface, or in parameters.xml.

- -security_username and -security_password
- -security_password_encrypted

Do not specify a value for *both* the -security_password and the -security_password_encrypted parameters.

Unless the super user password is modified with the modify_admin_password command, use dmcl_admin as the value for the -security_username and -security_password parameters.

For example, a Data Mover user who wants to run the start command could type:

```
datamove start -job_name PROD1_TO_DEV1 -security_username dmcl_admin
-security_password 53cUr17y
```

Alternatively the user could specify values for the security management elements in an XML file rather than passing the parameters on the command line. The security parameters must be the first elements that you specify in the parameters.xml file. The following parameters.xml file runs the start command for job PROD1_TO_DEV1, assumes that security management is enabled, and uses dmcl_admin as the user name for the super user and 53cUr17y as the password for the super user. If security management is enabled, always specify values for <security_username> and <security_password> elements immediately after the first element in the XML file.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<dmEdit
  xmlns="http://schemas.teradata.com/dataMover/v2009"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```

    xsi:schemaLocation="http://schemas.teradata.com/unity/datamover.xsd">
    <security_username>dmcl_admin</security_username>
    <security_password>53cUr17y</security_password>
    <job_name>PROD1_TO_DEV1</job_name>
  </dmEdit>

```

About Enabling CLI and TPT Trace Logs for TPT Jobs

Data Mover provides the ability to enable CLI tracing or TPT tracing at job level. Enabling CLI or TPT trace logging is not recommended unless needed; once enabled, all jobs begin generating extra trace logs.

Define the following elements in the job XML after the `query_band` element:

```

<enable_trace_log>
  <cli_trace_log>
    <netrace>21</netrace>
    <netrace_buf_len>0</netrace_buf_len>
  </cli_trace_log>
  <tpt_trace_log>
    <tptapi_debug>1</tptapi_debug>
  </tpt_trace_log>
</enable_trace_log>

```

Usage Notes

- When `cli_trace_log` elements are provided, CLI trace logs are enabled regardless of being a TPT job.
- Values greater than or equal to 0 for `netrace` and `netrace_buf_len` elements trigger CLI logs to generate. Must be valid `netrace` and `netrace_buf_len` values for CLI, Data Mover does not validate these values.
- When `tptapi_debug` is provided, the TPT trace log is enabled for TPT jobs; this setting does not affect non-TPT jobs. Must be a valid `tptapi_debug` value for Teradata PT API, Data Mover does not validate this value.
- Values greater than or equal to 0 for the `tptapi_debug` element triggers TPTAPI trace logs for TPT jobs.
- Values less than 0 for these elements does not trigger trace logs to generate.
- Both CLI and TPT trace logs are generated in the temp task directory and remain there if the job log level is 99 or the job fails; otherwise, the logs and task directory are removed after the job completes.

Job Definition Security

About Logging on to a Teradata Database System

When you use certain Data Mover commands, such as create, move, or start, you must log on to the source and target Teradata Databases defined for the job. To log on to a source or target Teradata Database system, provide at least one of the following to avoid errors during job creation:

- User name and password
- Logon mechanism and, if required, logon data

If both the user name, password, and the logon mechanism is valid, Data Mover connects to the source and target systems using the logon mechanism, username, and password. When only the logon mechanism is provided, Data Mover connects to the source and target systems using that logon mechanism.

Usage Notes

- Use logon mechanisms only for Teradata PT API and Teradata JDBC jobs.
- Logon mechanisms are not supported for Teradata DSA jobs.
- When using the create or move command, if **-source_logon_mechanism** or **-target_logon_mechanism** is specified and **-force_utility** is not used, Teradata PT API is used by default.
- Specifying **-source_logon_mechanism** or **-target_logon_mechanism** with Teradata DSA specified for **-force_utility** results in an error.

Password Security

For security purposes, it is recommended to not store passwords in a plain text file or specify passwords in parameters.xml. For example, the best practice is to specify the required passwords and parameters for logon mechanisms directly on the command line rather than in parameters.xml.

The following example uses parameters.xml to copy tables, but specifies passwords and parameters for logon mechanisms on the command line using the following parameters:

- Source password: 123456789
- Source logon mechanism parameter: joe@domain1@@mypassword
- Target password: 332675411
- Target logon mechanism parameter: margaret@domain2@@newpassword

```
datamove move -source_password 123456789
-source_logon_mechanism_data joe@domain1@@mypassword
-target_password 332675411 -target_logon_mechanism_data
margaret@domain2@@newpassword -f parameters.xml
```

Due to these security considerations, Data Mover does not require that passwords and parameters for logon mechanisms be included in `parameters.xml`, but does require that the information be supplied using the command line for those commands that require it. If you do not specify values for the required security parameters on the command line, you are prompted for the information.

Encrypting a Password

Password information is scrambled before being sent over the network and stored in the Data Mover database. To provide a greater level of security, you can also use the `encrypt_password` command.

The `encrypt_password` command creates an encrypted password from a password entered as a parameter on the command line, interactively, or through an XML file with a specific format. The generated password can then be used as a value for **-source_password_encrypted** or **-target_password_encrypted** when using the `create` and `move` commands.

1. Type `datamove encrypt_password` in the command line, and do one of the following:

Option	Description
Enter a password interactively	<ul style="list-style-type: none"> • Press Enter to be prompted to enter a password interactively. • Type the password you want to encrypt. Your input is masked with a set of asterisks.
Enter the password directly	<ul style="list-style-type: none"> • Type <code>-password</code> followed by the password you want to encrypt.
Enter the password by passing in an XML file	<ul style="list-style-type: none"> • Type <code>-f</code> followed by the name of the XML that passes in parameters for this command.

2. Press **Enter** to write the encrypted password to standard output, or type `-dir directory path` `-filename XML file name` where *XML file name* is a specified directory path and file name.

Using Multiple Data Mover Agents in Parallel

Jobs that copy large amounts of data using Teradata PT API can distribute the copy tasks for a single database object across multiple agents. Performance is improved because Data Mover draws on the combined resources of data streams associated with each agent. When an agent uses multiple data streams to process a task in parallel, the object is copied faster.

The following restrictions apply when using multiple agents in parallel:

- More than one agent must be installed in the Data Mover environment.
- The job can only use Teradata PT API.
- Jobs that use Teradata PT API only copy data in tables.

If Data Mover needs to perform additional SQL operations, this work is allocated to a single agent.

Note:

If `max_agents_per_task` is not specified or specified with no value, Data Mover dynamically sets the value for optimal performance.

The following parameters file for the create command assumes that the Data Mover environment has five online agents. Settings in this file are valid for jobs that use Teradata PT API.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<dmCreate xmlns="http://schemas.teradata.com/dataMover/v2009"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://schemas.teradata.com/unity/datamover.xsd">
    <job_name>floyd_dmdev_create</job_name>
    <source_tdpid>floyd</source_tdpid>

    <source_user>dmguest</source_user>
    <source_password>please</source_password>
    <target_tdpid>dmdev</target_tdpid>
    <target_user>dmguest</target_user>
    <target_password>please</target_password>
    <data_streams>4</data_streams>
    <max_agents_per_task>4</max_agents_per_task>
    <source_sessions>4</source_sessions>
    <target_sessions>4</target_sessions>
    <log_level>0</log_level>
    <database selection="unselected">
      <name>dmguest</name>
      <table selection="included">
        <name>orders_2010</name>
      </table>
    </database>
</dmCreate>
```

Using Multiple Agents for Jobs That Use Teradata PT API

Multiple agents can be used to copy a single large database object such as a table or a database with the create or move command for jobs that use Teradata PT API by editing the XML file for the command.

1. Add the `max_agents_per_task` element.
2. Set the value of the `max_agents_per_task` element to an integer number greater than one, but less than or equal to the number of installed, online agents.
This value corresponds to the maximum number of agents that are allocated for copying each database object in the job.
3. Set the value of the `data_streams` element to an integer number greater than 1.

Work is spread across multiple agents by assigning each agent one or more data streams.

4. For Teradata PT API only, set the value of `source_sessions` and `target_sessions` to an integer number greater than or equal to the value of `data_streams`.

Teradata PT API distributes sessions among the data streams. Because each Teradata PT API data stream requires at least one source and one target session, the number of source and target sessions must be greater than or equal to the number of data streams.

The parameter values for which all conditions must be **true** for Teradata PT API are:

Parameters	Product
<code>max_agents_per_task > 1</code>	Teradata PT API
<code>max_agents_per_task <= number-of-online-agents</code>	Teradata PT API
<code>data_streams > 1</code>	Teradata PT API
<code>source_sessions >= data_streams</code>	Teradata PT API
<code>target_sessions >= data_streams</code>	Teradata PT API

If you do not specify a value, Data Mover calculates `max_agents_per_task` dynamically for optimal performance.

The following table describes what occurs with different values for the `data_streams`, `max_agents_per_task`, `source_sessions`, and `target_sessions` parameters.

Value	Description
<code>max_agents_per_task</code> is greater than the number of online agents.	Teradata PT API: Sets <code>max_agents_per_task</code> equal to the number of online agents.
<code>data_streams</code> is greater than <code>max_agents_per_task</code>	Teradata PT API: Optimizes the distribution of the available data streams evenly among the online agents.
<code>max_agents_per_task</code> is greater than the value of <code>data_streams</code> .	The number of agents used for running a task is equal to the value of <code>data_streams</code> .
<code>data_streams</code> and <code>max_agents_per_task</code> are not provided.	Teradata PT API: Optimal Data streams is first computed, then sets the value of <code>max_agents_per_task</code> equal to the computed <code>data_streams</code> .
<code>source_sessions</code> or <code>target_sessions</code> is less than <code>data_streams</code> .	Teradata PT API only: Sets <code>source_sessions</code> or <code>target_sessions</code> equal to the number of data streams.
<code>source_sessions</code> or <code>target_sessions</code> are not provided.	Teradata PT API: Sets <code>source_sessions</code> or <code>target_sessions</code> dynamically to provide optimal performance.

Value	Description
data_streams is provided and max_agents_per_task is not provided.	Teradata PT API: Sets the value of max_agents_per_task equal to the data_streams.
data_streams is not provided and max_agents_per_task is provided.	Teradata PT API: Optimal data streams is computed. If max_agents_per_task is specified and the value is less than or equal to the number of computed data streams, the specified value is used; otherwise, the max_agents_per_task value is set to match number of computed data streams.

Using Event Tables

The following options are available for logging events:

- Log events directly in Teradata Ecosystem Manager.
- Log events to an internal table that Data Mover manages, referred to as an Event table.
- Log events both to Teradata Ecosystem Manager and to an Event table in Data Mover.

You can do the following when creating event tables:

- Set up one or more Event tables.
- Assign each Data Mover job to log its event messages to a particular event table by specifying the name of the event table in the job definition.
- A default event table can be chosen when no specific event table is set in the job definition.

Setting Up the Event Table

1. Run the create_event_table command in the command-line interface using parameters in the table:

Parameter	Example Value	Description
event_table_name	event1	Label that uniquely identifies the installation of the Event table.
system	tdsys1	Name of the Teradata system where the Event table will be installed. This can be the IP address or system alias.
user_name	dmuser1	Existing Teradata user for the Teradata system. User creates the Event table and inserts the events. Data Mover stores the user information. User must have CREATE and INSERT permissions on the specified event_database.
user_password	dmuser1pass	Password for the Teradata user associated with the user_name.

Parameter	Example Value	Description
event_database	db1	Name of the database on the Teradata server where the Event table is created.
use_existing_event_table	false	If an Event table already exists and this is set to true, Data Mover re-uses the existing table rather than create a new one.

Example:

```
datamove create_event_table -event_table_name event1 -system dmdev
-user_name user1 -user_password user1Pass -event_database db1
```

In the example, Data Mover creates table db1.TMSMEvent on the Teradata database system dmdev using Teradata user user1. This installation is referred to as event1.

The create_event_table command creates a table called TMSMEvent in the specified event_database. If the table already exists in the specified database and its schema is current, Data Mover re-uses the table. If no table exists, Data Mover creates a new one.

Setting Default Event Table to Log Events

To enable default logging for an Event table:

1. Set the tsm.mode configuration property to ONLY_INTERNAL_TSM or BOTH.
2. Set the event.table.default configuration property to the event_table_name of the Event table. Logs the events to the Event table for all Data Mover jobs where the user did not specify a value for log_to_event_table in the job definition.

Note:

You can also specify multiple default event tables in configuration.xml as follows:

```
<property>
  <key>event.table.default</key>
  <value>event1</value>
  <value>event2</value>
  <description>Purpose: When this property is set and tsm.mode is BOTH or
ONLY_INTERNAL_TSM, TSM messages will be saved to this event table, unless the
messages come from a different event table or the Job Definition explicitly
overrides this parameter. Default: NULL.</description>
</property>
```

Specifying the Event Table for a New Data Mover Job

1. Set the log_to_event_table parameter to event_table_name of the Event table.

Example:


```
<log_to_event_table>event1</log_to_event_table>
```

Specifying Multiple Event Tables for a New Data Mover Job

Data Mover enables you to send events for one job to multiple event tables; this capability provides additional security should any recovery process be needed. Multiple event tables are added by repeating the `log_to_event_table` tag in each job definition.

1. Set the `log_to_event_table` parameters to `event_table_name` for each event table.

Example:

```
<log_to_event_table>event1</log_to_event_table>
<log_to_event_table>event2</log_to_event_table>
```

Modifying an Event Table

The Teradata user name and password can be modified for an existing event table using the parameters listed in the following table.

Parameter	Example	Description
<code>event_table_name</code>	<code>event1</code>	Name that uniquely identifies the installation of the event table.
<code>change_user_name</code>	<code>true</code>	Determines whether or not the user name is changed.
<code>user_name</code>	<code>dmuser1</code>	Existing Teradata user name for Data Mover to access the event table. The user requires INSERT permission for the TMSMEvent table.
<code>change_user_password</code>	<code>true</code>	Determines whether or not the user password is changed.
<code>user_password</code>	<code>dmuser1pass</code>	Password for the Teradata user associated with the <code>user_name</code> .

1. Type the `modify_event_table` command using the parameters in the table.

```
datamove modify_event_table -event_table_name event1 -change_user_name true
-user_name dmuser2 -change_user_password true -user_password $131F
```

Deleting an Event Table

An event table can be deleted, using the `delete_event_table` command using the parameter in the following table:

Parameter	Example	Description
event_table_name	event1	Label that uniquely identifies the installation of the event table.

1. Type the `delete_event_table` command with the parameter `event_table_name` to delete an event table.

```
datamove delete_event_table -event_table_name event1
```

Listing All Existing Event Tables

All current existing event tables can be viewed using the `list_event_table` command. This command has no parameters.

1. Type the `list_event_table` command.

```
datamove list_event_table
```

Disabling Fallback

Data Mover allows you to disable fallback when copying objects between source and target systems. When source objects have FALLBACK enabled and the Data Mover job is replacing or creating the object on the target side. You have the following options when setting fallback in `list_configuration`:

Value	Description
<code>job.noFallback</code> is set to <code>true</code>	Fallback is disabled for the objects created in the target system.
<code>job.noFallback</code> is set to <code>false</code>	Objects created on the target system match the fallback status of the source objects.

Rules and Restrictions

- If the objects being copied already exist on the target system with FALLBACK enabled and `job.noFallback` is set to `true`, the NO FALLBACK setting is ignored. The exception is when running a DSA job, in that case, the setting is applied and the objects on the target system are copied with NO FALLBACK.
- If the `job.noFallback` setting is changed, any jobs using **freeze_job_steps** ignore the new setting until the job steps are manually updated.

Setting Up the QueryGrid T2T Utility

Data Mover supports the QueryGrid T2T utility. In order to use the QueryGrid T2T utility, both the source and target system must have the same QueryGrid version installed and meet the following minimum version requirements:

QueryGrid	Teradata Database	Data Mover
2.0	15.10.03	16.10

QueryGrid configuration on source, target, or both the system has the following rules:

- Create authorization and foreign server on the system where the link between source and target is available.
- Configure Proxy user and physical user on the system to which the link is pointing.

Note:

If the foreign server is defined on the target system, the link also resides on the target system, and designed to point from target to source. If the foreign server is defined on the source system, the link also resides on the source system, and designed to point from source to target.

1. To configure the foreign server on source, target, or both the systems, refer to:
 - [Configuring Foreign Server on Source System](#)
 - [Configuring Foreign Server on Target System](#)

Configuring Foreign Server on Source System

To configure the foreign server on the source system:

1. Follow the instructions in *Teradata® QueryGrid™ Installation and User Guide*, B035-5991 to configure QueryGrid.
 - a. Configure the proxy user and physical user on the target system to run T2T tasks, using the same physical user on the source system.
 - b. Connect the physical user to the proxy user on the target system.
 - c. Create an authorization and foreign server on the source system, ensuring that the source user has insert and select permissions on the foreign server.
2. Grant source user permissions on either `dbc.serverinfoV` or `dbc.serverInfoVX` – depending on your system view.
3. Logon to the source system as a source user and run the following two queries:
 - a. `Help foreign server <foreign_server_name>;`
 - b. `Select * from <db_name>.<table_name>@<foreign_server_name>;`

If you can run the queries without any issue, then the foreign server configuration completes successfully.
4. [Optional] Group pool is available with the T2T utility provided that all users are configured properly on both source and target systems. If you need to configure a group pool, follow these steps:
 - a. Create a list of the same physical users on both the source and target systems.
 - b. Create one proxy user on the target system and connect the physical user to the proxy user on the target system.

- c. Grant the physical users insert and select permissions to the foreign server on source system.
- d. Grant the physical users permissions on `dbc.serverinfoV` or `dbc.serverInfoVX` on source system.
- e. Add the source users to a group pool in the source system. The same set of users created on the target system do not necessarily need to be added to the group pool on target system.
5. Define a job using T2T as a `force_utility` using the `use_foreign_server` tag in the job definition. For more information, see [Creating a Job Using the T2T Utility](#).

Note:

If a `force_utility` is not specified, but a valid foreign server is, Data Mover automatically selects T2T to move the data. T2T is not selected if the job consists of entire databases being copied or the tables contain T2T data restrictions.

Configuring Foreign Server on Target System

To configure the foreign server on the target system:

1. Follow the instructions in *Teradata® QueryGrid™ Installation and User Guide*, B035-5991 to configure QueryGrid.
 - a. Configure the proxy user and physical user on the source system to run T2T tasks, using the same physical user on the target system.
 - b. Connect the physical user to the proxy user on the source system.
 - c. Create an authorization and foreign server on the target system, ensuring that the target user has insert and select permissions on the foreign server.
2. Grant target user permissions on either `dbc.serverinfoV` or `dbc.serverInfoVX` – depending on your system view.
3. Logon to the target system as a target user and run the following two queries:
 - a. `Help foreign server <foreign_server_name>;`
 - b. `Select * from <db_name>.<table_name>@<foreign_server_name>;`

If you can run the queries without any issue, then the foreign server configuration completes successfully.
4. [Optional] Group pool is available with the T2T utility provided that all users are configured properly on both source and target systems. If you need to configure a group pool, follow these steps:
 - a. Create a list of the same physical users on both the source and target systems.
 - b. Create one proxy user on the source system and connect the physical user to the proxy user on the source system.
 - c. Grant the physical users insert and select permissions to the foreign server on target system.
 - d. Grant the physical users permissions on `dbc.serverinfoV` or `dbc.serverInfoVX` on target system.

- e. Add the target users to a group pool in the target system. The same set of users created on the target system do not necessarily need to be added to the group pool on source system.
5. Define a job using T2T as a `force_utility` using the `use_foreign_server` tag in the job definition. For more information, see [Creating a Job Using the T2T Utility](#).

Note:

If a `force_utility` is not specified, but a valid foreign server is, Data Mover automatically selects T2T to move the data. T2T is not selected if the job consists of entire databases being copied or the tables contain T2T data restrictions.

Foreign Server Location Setup for T2T Jobs

The T2T jobs by default use the foreign server defined on source system. You can guide the T2T jobs to use the foreign server on target system by any of the following methods.

Note:

If the specified foreign server does not exist on the specified system you cannot create or run T2T jobs.

- In the job definition, add the **foreignServerOnTarget** attribute to the **use_foreign_server** element and set the value to `true`.

For example:

To use foreign server defined on source system:

```
<use_foreign_server>
  <name>myForeignServerOnSource</name>
</use_foreign_server>
```

To use foreign server defined on target system:

```
<use_foreign_server foreignServerOnTarget="true">
  <name>myForeignServerOnTarget</name>
</use_foreign_server>
```

- In the `configuration.xml` file, set the key `job.default.foreign.server.on.target` value to `true`. This setting forces all T2T jobs to use foreign server defined on target.

```
<property>
  <key>job.default.foreign.server.on.target</key>
  <value>true</value>
  <description>Purpose: If set to true Data Mover tries to utilize
foreign server on target system for T2T job. The default value is
```

```
false</description>
</property>
```

Changing the Foreign Server of T2T Jobs

You can update all the existing unfrozen T2T jobs from using foreign server on source to use the foreign server on target, without editing or re-creating the jobs.

1. Follow the instructions in *Teradata® QueryGrid™ Installation and User Guide*, B035-5991 to define a link from target to source on QueryGrid portlet.
2. Configure the foreign server on target system: [Configuring Foreign Server on Target System](#).
3. In the `configuration.xml` file, set the key `job.default.foreign.server.on.target` value to `true`.

Postrequisite:

Start an existing T2T job that was using the foreign server on source system. The job rebuilds and use the foreign server on target system.

DSA Utility Setup

Data Mover supports and comes bundled with the Teradata Data Stream Architecture (DSA) for copying databases and tables between Teradata systems.

The following components are part of the DSA features. It is important to note which components need installation and configuration before using the Data Mover DSA utility:

Component	Description
Data Stream Controller (DSC)	The DSC controls all backup and recovery (BAR) operations and is bundled and installed with the Data Mover daemon. You can use the Data Mover DSA utility with an external DSC or the bundled DSC. See the <i>Teradata® Data Mover Installation, Configuration, and Upgrade Guide for Customers</i> for more information.
DSMAIN	DSMAIN runs on Teradata Database nodes and receives job plans from the DSC. DSA MAIN must be configured and enabled before communicating with the Data Mover DSC to run jobs. DSA MAIN is installed on Teradata Database versions 14.10 and later. See the <i>Teradata® Data Mover Installation, Configuration, and Upgrade Guide for Customers</i> for information on configuring and enabling source and target systems to work with the Data Mover DSC.
DSA Network Client	Also known as BAR Network Client (referred to as either BAR NC or ClientHandler). The DSA Network Client copies data from the source DSA MAIN to the target DSA MAIN using shared network data pipes. Servers that have the DSA Network Client Software installed and configured are also known as Media Servers. It is recommended when using the Data Mover DSA utility that you chose either the source or target Teradata Database as the Media Server for optimal performance. See the <i>Teradata® Data</i>

Component	Description
	<i>Mover Installation, Configuration, and Upgrade Guide for Customers</i> for information on installing and configuring the DSA Network Client.
DSA Command-Line Interface (CLI)	The DSA command-line interface provides commands to define DSA configuration and is bundled and installed with the Data Mover Daemon.

Rules and Restrictions

- The Data Mover DSA utility requires additional configuration before use.
- If an external DSA designated for backup and restore application exists on your environment and you want to use the Data Mover DSA utility, you must upgrade the existing external DSA components (including DSC and BAR NC) to match the version of the Data Mover DSA utility.
- The following restrictions are true when a source or target system contains Teradata Database versions earlier than 16.00 and the Data Mover DSA utility is used to copy data:
 - The database does not work with more than one DSC environment at a time. The following table shows the errors that can occur when two DSCs use the same Teradata Database system that are versions earlier than 16.00:

Scenario	Result
DSA BAR environment already exists on system	A Data Mover environment with the bundled DSC is installed and the following jobs are run: <ol style="list-style-type: none"> 1. A BAR backup job from system A. 2. A Data Mover DSA job to copy data from System A to System B. If System A is using a database version earlier than 16.00, the BAR job no longer works. DSMAN on System A must be reconfigured when switching between DSC environments.
Two Data Mover environments exist: Test and Production	Both Data Mover environments have a bundled DSC and the following jobs are run: <ol style="list-style-type: none"> 1. A Data Mover DSA job in the Data Mover Test environment copying data from System A to System B. 2. A Data Mover DSA job in the Data Mover Production environment copying data from System B to System C. The job in Data Mover Test no longer works if System B is using a database version earlier than 16.00. If true, then DSMAN on System B must be reconfigured when switching between DSC environments.
Data Mover environment exists with automatic failover support consisting of an active and standby Data Mover daemon	Both the active and standby Data Mover daemons are configured with a bundled DSC and failover occurs after the following job is run: <ol style="list-style-type: none"> 1. Data Mover DSA job on the active Data Mover daemon copying data from System A to System B. The job on the newly promoted daemon fails if either database is using a version earlier than 16.00. DSMAN must be reconfigured when failover occurs on a database using a version prior 16.00.

- DSA is not automatically chosen for the source or target database when **force_utility** is not specified in the job definition. For advanced users that understand these restrictions and want to use the Data Mover DSA utility to copy data to earlier databases, you must specify DSA as the **force_utility**.

Using QueryGrid Manager to Report Job Status

QueryGrid Manager is a component of Teradata QueryGrid™ 2.0 that manages all QueryGrid systems. Data Mover uses the QueryGrid Manager REST service to report T2T progress. Configure the following properties in the `configuration.xml` file to view job status:

1. Enter either the `queryGridManagerPassword` or the `queryGridManagerEncryptedPassword` to access the QueryGrid Manager REST service.
2. Enter the `queryGridManagerUser` user.
The default user is `Support`. The username must be the same in QueryGrid Manager and Data Mover.
3. Set `queryGrid.wait.final.status` to `true` to wait until job status is returned from QueryGrid. Data Mover pulls the job status from QueryGrid Manager, but cannot always report the final status due to a delay from the QueryGrid REST service. When the final status is not available and `queryGrid.wait.final.status` is set to `false`, Data Mover displays the status as `N/A`. When `queryGrid.wait.final.status` is set to `true`, Data Mover continues to request the status until the final status is returned or a timeout occurs. This impacts job completion time.

```
<property>
  <key>queryGridManagerEncryptedPassword</key>
  <value>618f01fd9a1f0081ccb5ee4bc71d345fbf8b5ccb095f6bc0c89e390416622ee1</value>
  <description>Purpose: Set Querygrid Manager user encrypted password, cannot provided together with queryGridManagerPassword</description>
</property>
<property>
  <key>queryGridManagerPassword</key>
  <value></value>
  <description>Purpose: Set Querygrid Manager user password, cannot provided together with queryGridManagerEncryptedPassword</description>
</property>
<property>
  <key>queryGridManagerUser</key>
  <value>support</value>
  <description>Purpose: Set Querygrid Manager user, default value is support.</description>
</property>
<property>
  <key>querygrid.wait.final.status</key>
  <value>true</value>
```



```
<description>Purpose: Set if need to wait for Querygrid Manager to return
final task status. There will be performance impact when set to true.</
description>
</property>
```

Purging the Repository

Data Mover has an internal repository that stores job data from previously run jobs. The purge service is enabled by default and jobs are deleted every 60 days. You can adjust this setting and designate cleanup based on the following:

- Age of data
- Percent of the repository is full
- Both age and percent full

A daily cleanup check is performed at the time you specify; and if the age or space threshold is reached, the excess data is deleted from the repository. By default, Data Mover does not delete data that is newer than 5 days.

PostgreSQL allows all the memory available on the machine to be used as data storage, however, Data Mover considers the repository to be full at 100GB. Therefore, when using purge-by-percentage, the system compares the space used against the Data Mover maximum limit of 100GB.

1. Run `datamove list_configuration` to view the current Data Mover configuration settings.
2. Edit the following settings in the output `configuration.xml`:

Setting	Description
<code>repository.purge.enabled</code>	Set to TRUE to delete the job history.
<code>repository.purge.definition.enabled</code>	Set to TRUE to delete job history and job definitions.
<code>repository.purge.history.unit</code>	Set to days, weeks, months, or years to delete job history based on age.
<code>repository.purge.history.unitcount</code>	Set to number of days, weeks, months, or years to keep job history based on age. Job history older than this value is deleted. Set to -1 to disable delete by age.
<code>repository.purge.percent</code>	Enter a percentage of permanent space to keep available in the repository. Set to -1 to disable delete by percent.

3. Run `datamove save_configuration -f configuration.xml` to save configuration.

Job Management

You can use the Data Mover command line interface to perform many job management activities, including creating, running, stopping, and checking status of jobs. Although Data Mover automatically chooses a

default source and target session character set when creating or executing a job, you can manage this by selecting an available session character set supported by Teradata.

Creating a Job

Create a job on the daemon using the syntax parameters and the object list. You can specify job variable values directly on the command line, or in the XML file the command submits. If you specify different variable values for the same parameters in the command line and in the XML file, the value in the command line is used.

1. Type `datamove create -f objectlist.xml` and any job variable values that are not specified or that you want to override in the XML.
2. Make a note of the job name.
When the create command completes, the job name is displayed on the screen.

Note:

A job can also be created with the Move command, which does not require creating an object list first.

Related Information:

[Create command reference](#)

Running a Job

You can run a job as originally created, or you can specify new job variable values or an updated `parameters.xml` file at runtime or you can run a job which never exists with a set of job parameters from `commandline` or `parameters.xml`.

1. Type `datamove start -job_name PayableJob1`, where *PayableJob1* is the job name generated when the job was created, and continue with Step 2, if applicable.
2. [Optional] To make any changes in the original job definition, type the parameters followed by their new values or type `-f` followed by the XML file with the updated parameter values.
3. [Optional] To create and run a job which does exist before, type the set of parameters and values or type `-f` followed by the XML file. The parameters provided are used to create this job. If parameters

provided are not enough to create a job, the error message "Error: Job does not exist and not enough objects were selected to create a new job." is displayed.

Related Information:

[Start command reference](#)

Editing a Job

You can edit an existing job definition for a given job name using the same job parameters supported by the create command, except for the **job_name** and **table** parameters. If security management is enabled, you must meet the following conditions for the job:

- Have write permission
- Be either the job owner or be part of the Viewpoint administrator role

Note:

Spaces in the user or account name for the source or target ID cause jobs to fail.

Each job definition is associated with a job name. Each time you run a job using a particular job definition, you create an instance of the job with that name. If you run a job more than one time, you create multiple instances of the job based on that job definition. When you edit a job, you edit the underlying job definition, as well as all the instances of the job based on that job definition.

1. Type `datamove edit -job_name Job1` on the command line, where *Job1* is the job name displayed when the job was created, followed by the parameters with the necessary changes.
For example, typing `datamove edit -job_name PayableJob1 -log_level 99 -job_priority High` changes the level of logging to 99 and the execution priority for PayableJob1 to HIGH.

Note:

You can also supply new job variables by modifying the XML file and submitting it to the `edit` command, just as you can using the create command.

The following are considerations for security and modifying object names:

- If you are not the job owner and you want to change the existing job object list, you must enter credentials for the source and target systems.
- You can modify the list of objects being moved in a job using the XML file method only.
- When modifying any list of objects, you must supply the entire updated list of all objects in the job definition. If an object or object parameter specified in the original job definition is not included in the modified list, that object or object parameter is removed from the job definition. For example, if the created job has a list of tables and views and in the edited job only the view objects are specified, then the updated edited job will only have views specified.

- If no objects are listed, it is assumed that there is no update to the original object list.

Related Information:

[Edit command reference](#)

Checking Job Status

1. To get the job status of job, type `datamove status -job_name job_name` where *job_name* is the name of the job.

Related Information:

[Status command reference](#)

Stopping a Job

Use the stop command to stop a Data Mover job after the job has been started.

1. Type `datamove stop -job_name PayableJob1` on the command line, where *PayableJob1* is the name of the job.
The job status is set to USER CANCELLED. Agents are allowed to finish the job tasks already in progress, so that work on the job can continue after running the stop command. If the job was performing its final tasks, the job might complete. This allows the job to stop as cleanly as possible.
2. Once the job has stopped do one of the following:

Option	Description
Clean up the job using the <code>cleanup</code> command.	Release locks on the source and target system, and remove any temp tables or error tables (if created by the job).
Start the job again from the beginning using the <code>start</code> command.	Restarts the job.

Related Information:

[Stop command reference](#)

Cleaning Up a Job

Once a Data Mover job fails or is stopped with the stop command, the job might leave unwanted items on the source and target systems; including staging tables, error tables, and HUT locks. These remaining items can be removed with the `cleanup` command.

If the job used DSA, `cleanup` causes Data Mover to do the following:

- Stop any remaining DSA tasks that are still running.
- Release HUT locks from the source and target systems.

If the job used Teradata PT API or Teradata JDBC, `cleanup` causes Data Mover to drop error, work, log, and staging tables from the target system.

Note:

Use `start` to rerun the job from the beginning. The job cannot be restarted with `restart` after using the `cleanup` command.

Note the following considerations when cleaning up a job:

- Data Mover drops the target table during cleanup only if the table was created as a result of a failed or stopped Data Mover Teradata PT API job. The drop occurs because the Teradata PT API job won't complete successfully the next time it is executed unless the target table has already been dropped.
 - With partial copies, Data Mover does not delete any rows that might have been inserted or updated in the target table.
 - If the Teradata PT API Stream operator was used, macros that the Stream operator created are not automatically cleaned up. Instead, clean up the macros manually.
1. Type `cleanup -job_name PayableJob1` on the command line.
Where *PayableJob1* is the job name displayed when the job was created.

Related Information:

[Cleanup command reference](#)

Restarting Jobs

Data Mover provides table-level restarts. A table-level restart indicates that prior to the job failure, all of the table rows were copied successfully. Data Mover will not copy the table rows again if the job is restarted. However, if the job fails at any point while copying the rows for a table, Data Mover copies all of the table rows again when the job is restarted.

Data Mover jobs can only be restarted using the `restart` command under the following conditions:

- The job failed while running. The job has a status of `FAILED` when viewed with the `status` or `list_jobs` commands.
- The job was stopped with the `stop` command. The job has a status of `USER_CANCELLED` when viewed with the `status` or `list_jobs` commands.

In either case, do not run the `cleanup` command before restarting the job, or all work that the job did will be lost. Also, resolve whatever issued caused the job failure before restarting the job.

After the issue is resolved, you can restart the job with the `datamove restart -job_name <job name>` command.

Restart Actions

Data Mover takes different actions when restarting a job, depending on the step in which the job failed. Restart actions are described in the following table:

Step	Description	Data Mover Action During Restart
CLEAN_UP	Added to restart jobs. Tasks for cleaning up remnants of a failed job process.	Runs the entire restart job again.
MOVE_DEFINITION_BEFORE_LOAD or MOVE_DEF_AND_SRC_STG_DATA	Tasks for setting up staging tables.	Finishes performing setup tasks that were not completed, then runs the rest of the job.
MOVE_TABLE_DATA	Teradata PT API or Teradata JDBC tasks used to copy the table.	<ul style="list-style-type: none"> • Drops any remaining error, log, or work tables. • Drops and re-creates the staging table if the data had not been completely transferred. • Drops and re-creates the target table if all of the following conditions existed: <ul style="list-style-type: none"> ◦ A full-table copy was being performed. ◦ The target table was being directly loaded (instead of loaded to a staging table). ◦ The data had not been completely transferred to the target table. • Re-transfers all data for tables that did not have data completely transferred. • Runs the rest of the job.
RESOLVE_TABLE_AFTER_LOAD	Tasks such as performing INSERT or SELECT statements from staging to the target table, and adding indexes and triggers.	Completes tasks that were not completed, then runs the rest of the job.
COPY_STATISTICS	Tasks to copy statistics.	Completes tasks that were not completed, then runs the rest of the job.
ROW_COUNT_VALIDATION	Number of copied rows for each table, ensuring rows in source and target tables match. Normally the last step of a job.	Completes tasks that were not completed, then runs the rest of the job.

Additional Information

- The restart feature does not contain any logic to resolve the root issue of why the job failed. Resolve such issues before restarting the job.
- For jobs with failed DSA tasks, use `cleanup` instead of `restart`. DSA does not support `restart`.
- When performing restarts, Data Mover does not gather any new information about the source or target, but runs the remainder of the job as originally intended. Data Mover does not adapt the restart job to any changes that might have occurred since the job was originally created. In some cases, it might be necessary to run the `cleanup` command and create a new job if the original job plan is out of date.
- When a job is restarted, the `status` command output reflects only the status of the work being done for the restart job.

Making Job Steps Static

Each time a job is started, a verification step is performed to capture any changes in the source and target environments between the time the job was first created and the time it is run. This can negatively impact performance for jobs that run frequently.

For jobs with source and target environments that do not change after creation, use the **freeze_job_steps** parameter to prevent the job steps from being recreated each time the job is run.

If the **freeze_job_steps** parameter is true and the job target object status changes from non-exists to exists or from empty to non-empty, Data Mover recreates the job steps for only table and view objects. No rebuild is performed when using DSA.

If any changes occur in the source or target environments for the job, run the `update_job_steps` command to refresh the job steps.

1. Open the `parameters.xml` file containing the job definition information for the job.
2. Add the **freeze_job_steps** parameter in the appropriate place.
3. Set the value of **freeze_job_steps** to true.

If unspecified, the default value is false. If **freeze_job_steps** is true and the source or target environments change after job creation, an error may result when the job is run. In this case, run the `update_job_steps` command to refresh the job steps.

Updating Job Steps

When a saved job is executed, Data Mover performs a sequence of steps that includes verification of source and target environments.

Users who frequently run a job that does not need such verification may set the `freeze_job_steps` parameter to true to prevent job steps from being recreated each time the job is run.

If the **freeze_job_steps** parameter is true and the job target object status changes from non-exists to exists or from empty to non-empty, Data Mover recreates the job steps for only table and view objects. No rebuild is performed when using DSA.

If any changes occur in the source or target environments for the job, run the `update_job_steps` command to refresh the job steps.

1. At the Data Mover command line prompt, type `datamove update_job_steps -job_name` followed by the job name.

The job steps for the job you specified are updated without executing the job.

Related Information:

[Update Job Steps command reference](#)

Updating Job Priorities

You can update job priority for any job in the queue using only the `parameters.xml` file, not directly in the command line. For example, if jobs A and B are waiting in the queue and you want to run job B first, set the priority for B to HIGH and the priority for A to MEDIUM.

Note:

If security is enabled, the user's permissions must include **Change job priority** to modify the priority of jobs in the queue.

1. At the Data Mover command line prompt, type `datamove update_job_priorities -f parameters.xml`, where the `parameters.xml` file identifies the new priority for each job to be updated.

The output for `update_job_priorities` will contain two lists; one list identifies the jobs that were updated successfully (`success_list`), and one that identifies any jobs that failed with the reason (`failed_list`).

Related Information:

[Update Job Priorities command reference](#)

Default Sessions and Streams

Data Mover dynamically determines the number of source or target sessions and streams if you do not specify a value for `source_sessions`, `target_sessions` or `data_streams`. This determination is based on several factors, including the number of AMPs, number of AMPs in a map (if supporting the Teradata Database MAPS Architecture feature), size of objects being moved, and so forth. These dynamic values provide a good starting point if you are not familiar with tuning utilities to maximize performance. It is recommended that users test various combinations, as the ideal setting for each job will vary from site to site. In cases where a specific value better suits the requirements for a particular job, you can specify the value for `source_sessions`, or `target_sessions`, or `data_streams`.

The status output displays the number of source and target sessions or streams used to move each object.

For DSA jobs, `source_sessions` and `target_sessions` do not apply. DSA auto-determines the number of soft streams when `data_streams` is not specified. To limit the number of soft streams used during data transfer, specify a smaller `data_streams` value.

For TPT jobs, the number of sessions and streams is calculated for each object being moved. Every object being moved uses a different combination of the source and target sessions and data streams for optimal performance. If the source or target system supports MAPS, then the dynamic container or user job inputs for data streams and sessions must follow the following criteria:

- The source sessions cannot be greater than the AMP count in the source object map.

- The target sessions cannot be greater than the AMP count in the target object map.
- The data streams cannot be greater than the AMP count in the source or target object maps.

If the user provides a value for one or more of the three performance settings (`source_sessions`, or `target_sessions`, or `data_streams`), the unspecified values are automatically computed to work best with the user-specified values.

Run the `list_job_steps` command to view the sessions and streams that were dynamically computed by Data Mover.

Note:

If Teradata Active System Management (TASM) session rules are enabled on a particular source or target system, the TASM rules may override the Data Mover sessions.

About Specifying Source and Target Session Character Sets in a Job

Data Mover automatically chooses a default source and target session character set when creating or executing a job. The default session character set is ASCII, UTF8, or UTF16. The session character set chosen for each job is determined by the versions of the source and target systems and the characteristics of the objects being copied. Choosing a different source and target session character set is useful when the object names or data were originally created using a different session character set than any of the default values. For example, if data was originally inserted into the source table using KANJISJIS_0S as the session character set, you would want to specify KANJISJIS_0S as the source and target session character set when copying the data using Data Mover.

Data Mover supports specifying all of the session character sets supported by the Teradata JDBC driver as follows:

- ASCII
- UTF8
- UTF16
- KANJIEUC_0U
- KANJISJIS_0S
- HANGULKSC5601_2R4
- LATIN1_0A
- LATIN1252_0A
- LATIN9_0A
- SCHGB2312_1T0
- TCHBIG5_1R0

Data Mover does not support user defined session character sets or the 12 Teradata Database session character sets that the JDBC driver currently does not support. This is due to the fact that Data Mover cannot execute any queries on the source or target system if using any of these character sets.

The session character sets can be specified for both the source and target system in the job definition as follows:

```
<source_tdpid>hummingbird</source_tdpid>
<source_user>dmguest</source_user>
<source_password>dmguest</source_password>
<source_session_charset>KANJI EUC_0U</source_session_charset>
<target_tdpid>apollo</target_tdpid>
<target_user>dmguest</target_user>
<target_password>dmguest</target_password>
<target_session_charset>KANJI EUC_0U</target_session_charset>
```

Encoding for User Input

When a job request is submitted in XML or through a portlet, UTF-8 encoding is required for Data Mover to read the data correctly. Encoding sensitive data includes object names and where clauses. Data Mover stores this data in UTF-8 encoding in the internal repository. At run time, the text string is converted into the encoding determined by the session character set and sent to the underlying utilities for execution.

Support for Different Session Character Sets on the Source and Target System

When moving data from the source to a target system, use the same session character set to avoid data corruption. In certain conditions with Teradata PT API or JDBC, using different session character sets can convert the data from one encoding to another. Data Mover allows different source and target session character sets. A new default configuration parameter disables this feature. To create and execute jobs with a different source and target session character set, change the default value of this configuration parameter first. This guards against accidentally converting data when it is copied to the target system.



NOTICE

Use this feature with extreme care.

To enable this feature, set the property to true in the `save_configuration` command, as shown in the example here.

```
<property>
  <key>different.session.charsets.enabled</key>
  <value>true</value>
  <description>Purpose: Determines whether or not specifying different
source and target session character sets in a job is allowed. Default value false
```

```
means this is not allowed.</description>
</property>
```

When different session character sets are specified in a job definition, Data Mover chooses Teradata PT API over DSA when a value for the utility method is not specified. DSA does not really "convert" the data in this scenario and you may want the data converted if different source/target session character sets are specified in the job. If required, you can force Data Mover to use DSA.

Changing a Configuration Parameter

1. Run the `list_configuration` command to get the current configuration property settings.
An XML file is created with the current properties.
2. Edit the property value in the XML file. For example, `different.session.charsets.enabled` as `true`.
3. Run the `save_configuration` command to save the changes.

Teradata Database Session Character Sets Not Supported by Data Mover

The following table lists the twelve Teradata Database session character sets not currently supported by the JDBC driver and Data Mover.

Teradata Database Session Character Sets	Description
SCHINESE936_6R0	Windows Code Page 936 (Simplified Chinese, GB2312)
HANGUL949_7R0	Windows Code Page 949 (Korean KSC)
TCHINESE950_8R0	Windows Code Page 950 (Traditional Chinese, BIG5)
KANJI932_1S0	Windows Code Page 932 (Japanese Shift-JIS)
THAI874_4A0	Windows Code Page 874 (Thai)
LATIN1250_1A6	Windows Code Page 1250 (Czech, Croatian, Albanian, Hungarian, Polish, Romanian, Serbian, etc.)
CYRILLIC1251_2A0	Windows Code Page 1251 (Cyrillic, Russian)
LATIN1252_3A0	Windows Code Page 1252 (Latin 1)
HEBREW1255_5A0	Windows Code Page 1255 (Hebrew)
ARABIC1256_6A0	Windows Code Page 1256 (Arabic)
LATIN1254_7A0	Windows Code Page 1254 (Turkish)

Creating a Job Using the T2T Utility

You can create a T2T job by defining it as a **force_utility** and providing a foreign server tag in the job definition.

The following example shows T2T defined as the **force_utility**:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<dmCreate xmlns="http://schemas.teradata.com/dataMover/v2009"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://schemas.teradata.com/dataMover/
v2009/DataMover.xsd">
  <source_tdpid>sourceSys</source_tdpid>
  <source_user>source_user</source_user>
  <source_password>source_password</source_password>

  <target_tdpid>targetSys</target_tdpid>
  <target_user>target_user</target_user>
  <target_password>target_password</target_password>
  <data_streams>1</data_streams>
  <source_sessions>1</source_sessions>
  <target_sessions>1</target_sessions>
  <freeze_job_steps>FALSE</freeze_job_steps>
  <force_utility>T2T</force_utility>
  <use_foreign_server>
    <name>foreignservername</name>
  </use_foreign_server>
  <log_level>99</log_level>
  <database selection="unselected">
    <name>db1</name>
    <table selection="included">
      <name>tb1</name>
    </table>
  </database>
</dmCreate>
```

Creating a Job Using the DSA Utility

You can create and run jobs using the Data Mover DSA utility by setting the **force_utility** to DSA. When a **force_utility** is not specified, the Data Mover DSA utility is automatically selected when the Teradata Database is version 16.00 or later, provided that DSA is not a restricted case. See [DSA Utility Setup](#) before using the DSA utility.

When using DSA to copy data between systems, DSA preserves the block level compression. Data is only uncompressed on the target system if the target system has a different restore configuration (such as a different number of AMPs compared to the source system). If the target system has the same configuration as the source, the block level compression is preserved throughout the entire job. Since the data remains compressed in all scenarios when being transferred over the network, using DSA can result in significant performance improvement as compared to other copy methods when block level compression is used and the network is a limiting factor.

Rules and Restrictions

The Data Mover DSA utility has the following job restrictions:

- The DSA utility is available for Teradata Database versions 14.10 or later. However, DSA support for Teradata Database versions earlier than 16.00 is limited.
- DSA does not support moving journals.
- DSA does not support copying partial tables or view data unless source staging is specified.
- Failed DSA jobs cannot be restarted using the `restart` command. Use `clean up` followed by `start` to rerun the job after resolving the failure issue.
- When copying a database where either the source or target database version is earlier than 16.20, DSA does not preserve objects that exist only on the target database .
- DSA does not support copying tables and databases when indexes are included in the job and the source or target database version is earlier than 16.20.

DSA Optional Parameters

In job XML, an optional section can be added to support custom DSA parameters, as the following table describes:

Parameter Name	Required	Description
target_group_name	No	A target group is composed of Media Servers used for copying data. It is recommended that either the source or target system acts as a Media Server. When either the source or target system acts as a Media Server, it means the database nodes on that system have BAR NCs installed, configured, and running. Corresponding default target groups are automatically created by the Data Mover DSA utility and are used to copy data in the Data Mover job. This parameter should only be used by advanced users who want to specify a designated target group instead of using the default group created by the Data Mover DSA utility.
parallel_builds	No	The number of index and fallback sub-tables that can be built concurrently during a restore. The maximum number of concurrent builds is 5, default is 5.

The following example shows the optional DSA parameters added to the XML:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<dmCreate xmlns="http://schemas.teradata.com/dataMover/v2009"
```

```

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance
xsi:schemaLocation="http://schemas.teradata.com/dataMover/v2009/
DataMover.xsd">
  <source_tdpid>sourceSys</source_tdpid>
  <source_user>source_user</source_user>
  <source_password>source_password</source_password>
  <target_tdpid>targetSys</target_tdpid>
  <target_user>target_user</target_user>
  <target_password>target_password</target_password>
  <data_streams>1</data_streams>
  <source_sessions>1</source_sessions>
  <target_sessions>1</target_sessions>
  <freeze_job_steps>FALSE</freeze_job_steps>
  <force_utility>DSA</force_utility>
  <log_level>99</log_level>
  <dsa_options>
    <target_group_name>my_target_group</target_group_name>
    <parallel_builds>1</parallel_builds>
  </dsa_options>
  <database selection="unselected">
    <name>db1</name>
    <table selection="included">
      <name>tb1</name>
    </table>
  </database>
</dmCreate>

```

DSA Incremental Copy Jobs

Creating an Incremental Copy Job using DSA Utility

For information to create a DSA job, see [Creating a Job Using the DSA Utility](#). Previously, to keep a table in sync between two systems, it was important to create and run a full table copy, followed by creating or running multiple partial table copy jobs with appropriate SQL `where` clause.

Now you have the option of creating and running an Incremental Copy job using DSA utility:

- For the first execution, the job copies the table data in FULL
- For the subsequent executions, the job copies the incremental or DELTA changes from the source to the target

Rules and Restrictions

The Incremental Copy feature is supported by DSA utility, known as Increment Restore (IR) within DSA. The incremental copy job must already be eligible to run with DSA. For more information see the Rules and Restriction section at [Creating a Job Using the DSA Utility](#)

In addition, the following rules and restrictions apply:

Source and Target Systems

- The source and target systems must be 16.20.53.07 or beyond
- The source system must enable backup for IR:
 - Set GDO 553 – EnableBackupsforIncrementalRestore > 0

Enable BackupsforIncrementalRestore for object level

Enable BackupsforIncrementalRestore for whole system

- Restart bardsmain on the target database cnsterm 6:

Stop bardsmain: cnsterm 6 start bardsmain -s

Start bardsmain: cnsterm 6 start bardsmain

- The target system must enable Incremental Restore (IR)

GDO 555 - IgnoreIncrementalRestoreState = false

The GDO flag is set to false by default and no further action is needed.

If the flag has been modified to "true" previously, run `dsc reset_ir_target_system` command to clear the flag so IR is re-enabled.

Incremental Copy is supported in the following versions:

Source SQL-E Version	Target SQL-E Version			
	17.10.00.00	17.05.00.00	17.00.00.00	16.20.53.07+
17.10.00.00	Yes	—	—	—
17.05.00.00	Yes	Yes	—	—
17.00.00.00	Yes	Yes	Yes	—
16.20.53.07+	Yes	Yes (must be 17.05.05.03+)	—	Yes

Further job restrictions

The incremental copy job does not support copying the following objects:

- Join and hash indexes
- Partial data copy
- Job that needs source or target staging tables in either job definition or job steps

User cannot change an existing job to an incremental copy job and vice versa

Incremental Copy Parameters

To enable incremental copy job with the Data Mover command line, add XML tag `enable_incremental_restore` and set to "TRUE" after `<log_to_event_table>` xml tag.

There are a couple of optional parameters under `<dsa_options>` tag that must be left as UNSPECIFIED. (See [Limitations and Caveats](#) and [Troubleshooting DELTA Execution Failures](#) sections on when to apply these parameters):

Parameter Name	Required	Description
<code>ir_allow_write</code>	No	Used when incremental restore is enabled. Allow write for objects after incremental restore. The valid values are: UNSPECIFIED, TRUE, or FALSE.
<code>ir_execution_type</code>	No	Used when incremental restore is enabled. Incremental restore execution type. The valid values are: UNSPECIFIED or FULL.

The following example shows all the parameters added to the XML for a DSA incremental copy job:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<dmCreate xmlns="http://schemas.teradata.com/dataMover/v2009" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://
schemas.teradata.com/unity/DataMover.xsd">
  <source_tdpid>sourceSys</source_tdpid>
  <source_user>source_user</source_user>
  <source_password>source_password</source_password>
  <target_tdpid>targetSys</target_tdpid>
  <target_user>target_user</target_user>
  <target_password>target_password</target_password>
  <use_userid_pool>>false</use_userid_pool>
  <overwrite_existing_objects>TRUE</overwrite_existing_objects>
  <freeze_job_steps>UNSPECIFIED</freeze_job_steps>
  <force_utility>DSA</force_utility>
  <compare_ddl>UNSPECIFIED</compare_ddl>
  <log_level>99</log_level>
  <online_archive>UNSPECIFIED</online_archive>
  <log_to_event_table>tmsmevent</log_to_event_table>
  <enable_incremental_restore>TRUE</enable_incremental_restore>
  <dsa_options>
```



```

    <ir_allow_write>UNSPECIFIED</ir_allow_write>
    <ir_execution_type>UNSPECIFIED</ir_execution_type>
  </dsa_options>
  <db_client_encryption>UNSPECIFIED</db_client_encryption>
  <database selection="unselected">
    <name>db1</name>
    <table selection="included">
      <name>tb1</name>
    </table>
  </database>
</dmCreate>

```

Limitations and Caveats

Performance Limitation

DSA Incremental Copy is not the same as change data capture (CDC) because DSA is a block copying utility.

During DELTA executions, DSA copies change data blocks from source to target. Hence, even when the rate of data change is small during DELTA copy, the expected performance gain may not be observed from FULL copy, if changed rows affects a large number of underlying data blocks.

Read-Only Target

To make sure that the source and target data are in sync for DELTA executions during incremental copy, DSA sets the target table to read-only after incremental FULL execution, also known as the incremental restore state.

User is not allowed to change target table data, including adding join or hash indexes, which results in Operation not allowed: table is in Incremental Restore (read-only) state.

- If target system must be promoted during an emergency, such as a DR system standing in for a PROD system, user can set GDO 555IgnoreIncrementalRestoreState to true to make the entire system write enabled
- If user only wants to make the target tables in an incremental job write enabled, user can start the job with parameter -ir_allow_write set to "true": `datamove start -job_name my_ir_job -ir_allow_write true`
- Invoke a SQL statement to allow write for the read-only object:

The syntax is:

INCREMENTAL RESTORE ALLOW WRITE FOR

| --<Databasename> -- |

```
| --<Databasename.TableName> --| ;
```

Dynamically Forced FULL Executions

Ideally, a DSA incremental copy job only starts a one-time FULL copy during the first job run, with DELTA executions to follow.

DSA dynamically forces a FULL execution if any of the following happens after a previous job execution:

- A table is write enabled, for example, if user started the job with "-ir_allow_write true" in the previous job run
- Source or target incremental restore GDO flags have changed
- User has run "dsc reset_ir_target_system" command to reset target system GDO flag to IR enabled
- Job definition has changed affecting job objects or job options including online archive, skip stats, and skip join or hash indexes.

Troubleshooting DELTA Execution Failures

During DELTA executions, previously successful incremental copy jobs may fail with following errors:

- Error code: 1187, error message: Incremental Restore is not allowed because table 'db.table' is not in read-only state
- Error code: 1186, error message: Incremental Restore is not being processed in the correct order for table 'db.table'

These errors indicate job conflicts due to multiple jobs copying to the same target table, for example:

- A BAR restore job and a Data Mover incremental job copying to the same target table
- Two Data Mover incremental jobs copying to the same target table

To recover from the job failures, re-run the failed incremental job with "ir_execution_type" parameter set to "full" to do a FULL table copy: `datamove start -job_name my_ir_job -ir_execution_type full`

To prevent future errors, user needs to make sure not to restore or copy to the same target table across BAR and Data Mover DSCs.

Incremental Restore Command Parameters

For existing parameters, see, [About Data Mover Commands](#)

Additional parameters for "create", "edit", "move", and "start" commands:

<code>ir_allow_write</code>	UNSPECIFIED	(optional) Allow write for objects after incremental restore.
<code>ir_execution_type</code>	UNSPECIFIED	(optional) set DSA job

incremental restore execution type. The valid values are: FULL or UNSPECIFIED.
 enable_incremental_restore UNSPECIFIED (optional) indicates if this job is enabled to copy data using DSA incremental restore.

Incremental Restore Rest Parameters

For information on RESTful API settings type, see, [SettingsType](#)

New parameter:

Name	Description	JSON Data Type	Required
enableIncrementalRestore	Enable for incremental copy	String: TRUE, FALSE, UNSPECIFIED	No

For information on DSA settings type, see, [DSASettingsType](#)

New parameter:

Name	JASON Data Type	Required
irAllowWrite	String: TRUE, FALSE, UNSPECIFIED	No
irExecutionType	String: FULL, UNSPECIFIED	No

For example on Teradata to Teradata REST example for sample job JASON, see, [Teradata to Teradata REST Example](#)

Following is a JSON snippet showing new IR settings:

```
"settings":
{
  "priority": "MEDIUM",
  "overwriteExistingObjects": "true",
  "freezeJobSteps": "false",
  "targetDatabase": "targetDatabaseJobLevel",
  "compareDDL": "true",
  "logLevel": "99",
  "map" : "mapname1",
  "enableIncrementalRestore": "FALSE",
  "tdTdSettings":
  {
    "forceUtility": "DSA",
    "onlineArchive": "false"
  },
  "dsaSettings":{
```

```

        "parallelBuilds": 5,
        "irAllowWrite": "UNSPECIFIED",
        "irExecutionType": "UNSPECIFIED"
    },
    },

```

DSA Cloud Staging Copy Jobs

The Cloud Staging Copy Service feature provides a method of copying data where data is first backed up from the source system to cloud storage and then copied from cloud storage to the target system. This feature uses the ability of DSA to back up to and restore from cloud storage. The cloud storage space where data is temporarily stored is called the cloud staging area. After copying data to the target system, the Cloud Staging Copy Service removes the data from the cloud staging area.

Note:

Currently the Data Mover supports only AWS S3 cloud storage.

Creating a Cloud Staging Area

Cloud Staging Copy Service uses a cloud staging area to copy data between systems. You need to specify the cloud staging area when creating a Data Mover job. To create a cloud staging area:

1. Update the `cloudstagingarea.xml` file with the details as mentioned in the following topics.
The requirements for the `cloudstagingarea.xml` depend upon the method you are using to create the cloud staging area.
 - [User Defined Target Groups](#)
 - [Cloud Staging Copy Service Defined Target Groups](#)
2. Run `datamove create_cloud_staging -f cloudstagingarea.xml`.
When the command completes, a message appears stating that the cloud staging area created successfully, or an error report appears, if occurred.

User Defined Target Groups

User defined target groups are where you have already created the necessary configuration on Data Mover DSC to allow backup/restore to/from S3. If the configuration is already setup, then you only have to provide the source system target group (target group for backup) and target system target group (target group for restore) in the `cloudstagingarea.xml` file.

The necessary configurations on the Data Mover DSC are:

- Creating AWS S3 account
- Creating two target groups (one for backup and one for restore) and linking to the AWS S3 account
- Creating a target group map that links the backup and restore target group

A sample `cloudstagingarea.xml` file for creating a cloud staging area with user defined target groups is:

```
{
  "name": "cs2-cloudstagingarea",
  "storage_type": "S3",
  "source_target_pairs": [
    {
      "source_system": "sourceSystemA",
      "source_system_target_group": "my_source_target_groupA",
      "target_system": "targetSystemB",
      "target_system_target_group": "my_target_target_groupB"
    },
    {
      "source_system": "sourceSystemA",
      "source_system_target_group": "my_source_target_groupC",
      "target_system": "sourceSystemC",
      "target_system_target_group": "my_target_target_groupD"
    }
  ]
}
```

You need to provide the following details in the `cloudstagingarea.xml` file.

- **source_system_target_group**
- **target_system_target_group**

When you run the command to create cloud staging area, the Cloud Staging Copy Service validates the **source_system_target_group** and **target_system_target_group** for each source and target pair, as following:

- The target groups exist.
- The target group map is linking the two target groups.
- The AWS/buckets specified in the target groups are same.

Note:

The AWS/regions/buckets and **prefix_list** need to match between the two target groups in each source target pair.

Cloud Staging Copy Service Defined Target Groups

Cloud Staging Copy Service can create all the necessary DSA configuration for each source/target pair to run backup/restore jobs using S3.

A sample `cloudstagingarea.xml` file for creating a cloud staging area by Cloud Staging Copy Service defined target groups is:

```

{
  "name": "cs2-cloudstagingarea",
  "storage_type": "S3",
  "s3_properties": {
    "access_key_id": "ABCDEFGH",
    "secret_access_key": "AbcDEfgHIjklmNop123/456qRstUVwXyZ",
    "buckets_by_region": [
      {
        "buckets": [
          {
            "bucket_name": "example-bucket",
            "prefix_list": [
              {
                "prefix_name": "backup",
                "storage_devices": 100
              }
            ]
          }
        ]
      },
      {
        "region": "us-west-2"
      }
    ]
  },
  "source_target_pairs": [
    {
      "source_system": "sourceSystemA",
      "target_system": "targetSystemB"
    },
    {
      "source_system": "sourceSystemA",
      "target_system": "sourceSystemC"
    }
  ]
}

```

You need to provide the following details in the `cloudstagingarea.xml` file.

- **s3_properties** which contains s3 information as well as the region/bucket/prefix_list.
- **access_key_id**
- **secret_access_key**

Restrictions for Using the Cloud Staging Copy Service Defined Target Groups

Following are the restrictions for a cloud staging area created by Cloud Staging Copy Service defined target groups:

- You need to provide the **access_key_id** and **secret_access_key** for authentication. The process does not support any other authentication methods.
- In **s3_properties** you can provide only one value for each of the following parameters:
 - **buckets_by_region**
 - **bucket_name**
 - **prefix_list**
- You cannot specify a **source_system_target_group** or **target_system_target_group** inside a source/target pair.
- Cloud Staging Copy Service creates the target groups using the media servers listed in **default shared target group** for the Teradata system specified in the source/target pair. If a default shared target group does not exist, then Cloud Staging Copy Service cannot create a cloud staging area.
- When you start the configuration, the Cloud Staging Copy Service always attempts to create a new S3 AWS account in DSC, and does not reuse the existing account. If the **access_key_id** of the existing account matches with the **access_key_id** in new configuration, the configuration error occurs.

Editing a Cloud Staging Area

Following are the restrictions in editing a cloud staging area:

- You cannot edit a cloud staging area when a job linked to that cloud staging area is running.
- You cannot update the Cloud Staging Area name and the Buckets by Region.
- If cloud staging area is created by manually defining target groups, you can modify only source/target pairs.
- If cloud staging area is created by Cloud Staging Copy Service defined target groups, you can modify only the Access Key ID and the Secret Access Key.

Note:

To update the Cloud Staging Area name and Buckets by Region, you need to delete and create the cloud staging area again.

To edit the cloud staging area:

1. Update the `cloudstagingarea_edit.xml` file with the desired changes in the cloud staging area.
2. Run `datamove edit_cloud_staging -f cloudstagingarea_edit.xml`.

When the command completes, a message appears stating that the cloud staging area updated successfully, or an error report appears, if occurred.

Getting Details of a Cloud Staging Area

You can retrieve details of a specific cloud staging area directly using the command line parameter or using the `getcloudstaging.xml` file.

1. To run `get_cloud_staging` you can type either of the following:

- For command line parameter:

```
datamove get_cloud_staging -name example_area -filename output.xml
```

- For using xml file:

```
datamove get_cloud_staging -f getcloudstaging.xml
```

2. Mention the parameters/xml properties as following:

- For command line parameter, replace the `example_area` with the cloud staging area name and the `output.xml` with the output file name.
- For xml file, specify the output file name in the **<file>** property of the `getcloudstaging.xml` file.

3. Run the script.

When the command completes, the details of the cloud staging area appears in the output file. If the output file is not specified, the default behavior is to use the cloud staging area name as the output filename.

Listing Cloud Staging Area

To get a list of all cloud staging areas:

1. Run `datamove list_cloud_staging`.

When the command completes, a table appears with the name, source, and target information for each cloud staging area, or an error report appears, if occurred.

Deleting a Cloud Staging Area

Prerequisite:

Make sure, no Data Mover cloud staging copy job is using or linked with the cloud staging area.

You can delete a cloud staging area directly using the command line parameter or using the `deletecloudstaging.xml` file.

1. To run `delete_cloud_staging` you can type either of the following:

- For command line parameter:

```
datamove delete_cloud_staging -name example_area
```

- For using xml file:

```
delete_cloud_staging -f deletecloudstaging.xml
```

2. For command line parameter, replace the `example_area` with the cloud staging area name.
3. Run the script.

When the command completes, a message appears stating that the cloud staging area deleted successfully, or an error report appears, if occurred.

Creating a Data Mover Cloud Staging Copy Job

Prerequisite:

Create a valid cloud staging area before creating a Data Mover Cloud Staging Copy Job.

Incremental copy method is not yet added as a feature for cloud staging copy jobs. Use only Full and Partial copy methods.

To Create a Data Mover Cloud Staging Copy Job:

1. Add the cloud staging area in the `cloudstagingjob.xml` file.
2. Run `datamove create -f cloudstagingjob.xml`.

Working with Staging Databases

Work with staging databases by specifying staging at the following levels:

- Database and table levels
- Secondary staging database
- System level
- Job level
- Database level
- Object level

Note:

Target staging tables are created as NOPI tables unless `TPTAPI_UPDATE` is being used to load data to the staging table. `TPTAPI_UPDATE` does not support loading to a NOPI staging table. In this case, if the table is a time series table, the target staging table is created with the default Primary Index; otherwise, the target staging table is created with the same Primary Index as the source table. Source staging tables for a table object are created using the original source table ddl. Source staging tables for a view object are created the same way as the target staging table.

Specified staging databases are only used when the job needs to create temporary objects. The following table lists when a staging database is used, based on the force utility selected.

Force Utility	When Target Staging Database is Used
DSA	Table is a child on the source database and does not exist on the target database.
	Table exists on the target database and has either a parent or child on the target database.

Force Utility	When Target Staging Database is Used
	Job is a partial copy, source staging is used, and the target table is not empty.
	Target table exists and <code>force_target_staging_table</code> is true.
	Job is a partial copy of view data directly to a target view when source staging is used.
	Table has a trigger with a before action.
JDBC	Target table exists and is not empty.
	Target table is an RI parent.
	Target table is an RI child.
	Target table has indexes.
	Table is a temporal table.
	Target table has secondary indexes.
	Job contains a trigger with action time before.
	Target table has a trigger.
	Target table exists and is empty, but <code>force_target_staging_table</code> is true.
	Job is copying view data to target view.
T2T	Target table exists and is not empty.
	Target table is an RI parent.
	Target table is an RI child.
	Target table has indexes.
	Table is a temporal table.
	Target table has secondary indexes.
	Job contains a trigger with action time before.
	Target table has a trigger.
	Target table exists and is empty, but <code>force_target_staging_table</code> is true.
	Job is copying view data to target view.
TPT	Target table is not empty and is not performing an upsert.
	Target table is an RI child and is not performing an upsert.
	Target table has indexes and not performing an upsert.
	Table is a temporal table and is performing an upsert.
	Target table has secondary indexes and is not performing an upsert.

Force Utility	When Target Staging Database is Used
	Job contains a trigger with action time before and is not performing an upsert.
	Target table contains a trigger and is not performing an upsert.
	Target table exists and is empty, but <code>force_target_staging_table</code> is true.
	Job is copying view data to target view.

Specifying Target Staging at the Database and Table Levels

You can specify a target staging database name in the create or copy XML file, at the database level or the table level. A separate database for staging tables, macros, work tables, error tables, and log tables is useful if, for example, there is not enough space on the database where the target tables are located. This feature is optional and valid when using Teradata DSA, Teradata PT API, or Teradata JDBC.

Note:

All target staging tables, macros, work tables, error tables, and log tables are created based on the database specification.

Database Specification	Result
Not provided at the table or database level	Are created in the target database of the target tables.
Provided at the database level	Are created under the specified target staging database on the target system, unless overridden at the table level.
Provided at the table level	Are created under the specified target staging database on the target system, even if a different target staging database name is provided at the database level.

Example: Specifying Target Staging at the Database and Table Levels

In this example, the `f_table1` table uses the `staging_db_two` database. The `staging_db_one` database specified at the database level is overridden. The `f_table2` table uses the `staging_db_one` database that is specified at the database level. Data Mover supports using either the `staging_database` or `target_staging_database` tags to specify the target staging database, but not both at the same time.

```
<database selection="unselected">
  <name>DBName</name>
  <staging_database>
    <name>staging_db_one</name>
  </staging_database>
```

```

    <table selection="included">
      <name>f_table1</name>
      <staging_database>
        <name>staging_db_two</name>
      </staging_database>
    </table>
    <table selection="included">
      <name>f_table2</name>
    </table>
  </database>

```

Example: Specifying Target Staging at Database Level Only

In this example, the `f_table1` table uses `staging_db_two` as its staging database. However, for the `f_table2` table, a staging database name is not specified at the table level. Therefore, the table uses the target table database, `MyDB`, as the staging database. Data Mover supports using either the `staging_database` or `target_staging_database` tags to specify the target staging database, but not both at the same time.

```

<database selection="unselected">
  <name>MyDB</name>
  <table selection="included">
    <name>f_table1</name>
    <staging_database>
      <name>staging_db_two</name>
    </staging_database>
  </table>
  <table selection="included">
    <name>f_table2</name>
  </table>
</database>

```

Example: Specifying Target Staging at Table Level Only

In this example, both tables use `staging_db_one` as their staging database because the staging database name is specified only at the database level. Data Mover supports using either the `staging_database` or `target_staging_database` tags to specify the target staging database, but not both at the same time.

```

<database selection="unselected">
  <name>MyDB</name>
  <staging_database>
    <name>staging_db_one</name>
  </staging_database>

```

```

    <table selection="included">
      <name>f_table1</name>
    </table>
    <table selection="included">
      <name>f_table2</name>
    </table>
  </database>

```

Specifying a Secondary Target Staging Database

If you specify a staging database, the staging database by default contains the staging target table and any work tables, log tables, error tables, or macros that need to be created. You may want the staging target table to be created in a staging database that is separate from the staging database where the work tables, log tables, error tables, and macros are created.

You can specify a secondary staging database to hold only the staging target table by using the optional `staging_database_for_table` tag in the job creation XML file. If the value for the `staging_database_for_table` tag is specified, that database is used as the staging database for the staging target table. The `staging_database` tag is still used for specifying the general staging database in which the work tables, log tables, error tables, and macros are stored.

Similar to the `staging_database` tag, the `staging_database_for_table` tag can be applied either at the database level or at the table or view level. The value specified at the table or view level overwrites the value specified at the database level.

Data Mover supports using either the `staging_database` or `target_staging_database` tags to specify the target staging database, but not both at the same time.

Example: Specifying General and Secondary Staging Databases at Database and Table Levels

In this example, for table `test1` the staging table is created in staging database `staging4`, and the work, log, and error tables are created in staging database `staging3`.

```

<database selection="unselected">
  <name>database1</name>
  <staging_database><name>staging1</name></staging_database>
  <staging_database_for_table><name>staging2</name></
staging_database_for_table>
  <table selection="included">
    <name>test1</name>
    <staging_database><name>staging3</name></staging_database>
    <staging_database_for_table><name>staging4</name></
staging_database_for_table>

```

```

    </table>
</database>

```

Example: Specifying a Secondary Staging Database at Database Level and General Staging Database at Table Level

In this example, for table test1 the staging table is created in staging database staging2, and the work, log, and error tables are created in staging database staging3.

```

<database selection="unselected">
  <name>database1</name>
  <staging_database><name>staging1</name></staging_database>
  <staging_database_for_table><name>staging2</name></
staging_database_for_table>
  <table selection="included">
    <name>test1</name>
    <staging_database><name>staging3</name></staging_database>
  </table>
</database>

```

Example: Specifying Secondary Staging Database at Database Level

In this example, for table test1 the staging table is created in staging database staging2, and the work, log, and error tables are created in staging database staging1.

```

<database selection="unselected">
  <name>database1</name>
  <staging_database><name>staging1</name></staging_database>
  <staging_database_for_table><name>staging2</name></
staging_database_for_table>
  <table selection="included">
    <name>test1</name>
  </table>
</database>

```

Example: Specifying Secondary Staging Databases at Database and Table Levels

In this example, for table test1 the staging table is created in staging database staging4, and the work, log, and error tables are created in staging database staging1.

```

<database selection="unselected">
  <name>database1</name>
  <staging_database><name>staging1</name></staging_database>
  <staging_database_for_table><name>staging2</name></
staging_database_for_table>
  <table selection="included">
    <name>test1</name>
    <staging_database_for_table><name>staging4</name></
staging_database_for_table>
  </table>
</database>

```

Example: Specifying General Staging Databases at Database and Table Levels

In this example, for table test1 the staging table and the work, log, and error tables are all created in staging database staging3.

```

<database selection="unselected">
  <name>database1</name>
  <staging_database><name>staging1</name></staging_database>
  <table selection="included">
    <name>test1</name>
    <staging_database><name>staging3</name></staging_database>
  </table>
</database>

```

Specifying Source Staging at the Database and Table Levels

You can use DSA to copy partial data from a table or view using a source staging table. Specify where to create the source staging table by specifying a source staging database.

Staging Database Name	Staging Database Name Result
Not provided at the table or database level	Created in the source database of the source tables.
Provided at the database level	Created under the specified staging database on the source system, unless overridden at the table level.
Provided at the table level	Are created under the specified staging database on the source system, even if a different staging database name is provided.

Example: Specifying Source Staging Database Name at Database Level and Table Level

In this example, table `f_table1` uses the `staging_db_two` database. The database specified at the database level, `staging_db_one`, is overridden. The table `f_table2` uses the `staging_db_one` database that is specified at the database level.

```
<database selection="unselected">
  <name>DBName</name>
  <source_staging_database>
    <name>staging_db_one</name>
  </source_staging_database>
  <table selection="included">
    <name>f_table1</name>
    <use_source_staging_table>TRUE</use_source_staging_table>
    <source_staging_database>
      <name>staging_db_two</name>
    </source_staging_database>
  </table>
  <table selection="included">
    <name>f_table2</name>
    <use_source_staging_table>TRUE</use_source_staging_table>
  </table>
</database>
```

Example: Specifying Source Staging Database Name at Database Level Only

In this example, table `f_table1` uses `staging_db_two` as its source staging database and table `f_table2` does not specify the staging database name at the table level. This results in the table using the source table database, `MyDB`, as the source staging database.

```
<database selection="unselected">
  <name>MyDB</name>
  <table selection="included">
    <name>f_table1</name>
    <use_source_staging_table>TRUE</use_source_staging_table>
    <source_staging_database>
      <name>staging_db_two</name>
    </source_staging_database>
  </table>
  <table selection="included">
```



```

        <name>f_table2</name>
        <use_source_staging_table>TRUE</use_source_staging_table>
    </table>
</database>

```

Example: Specifying Source Staging Database Name at Table Level Only

In this example, the staging database name is specified only at the database level. This results in both tables use staging_db_one as their staging database.

```

<database selection="unselected">
    <name>MyDB</name>
    <source_staging_database>
    <name>staging_db_one</name>
    </source_staging_database >
    <table selection="included">
        <name>f_table1</name>
        <use_source_staging_table>TRUE</use_source_staging_table>
    </table>
    <table selection="included">
        <name>f_table2</name>
        <use_source_staging_table>TRUE</use_source_staging_table>
    </table>
</database>

```

Specifying Staging and Target at the Job and System Levels

During job creation, the following tags are supported at the job level:

- source_staging_database
- target_staging_database or staging_database
- staging_database_for_table
- target_database

Data Mover supports using either the staging_database or target_staging_database tags to specify the target staging database, but not both at the same time.

Specified values are applied to all the table objects included in the job request. These tags are optional and must be positioned after the force_utility tag.

```

<force_utility>tptapi</force_utility>
<source_staging_database><name>stagingDB</name></source_staging_database>
<!--target_staging_database><name>stagingDB</name></target_staging_database-->
<staging_database><name>stagingDB</name></staging_database>

```

```
<staging_database_for_table><name>stagingDBForTable</name></staging_database_for_table>
```

Job Level Specification Using Job Creation XML

Job Creation XML supports the following tags at the job level:

- `source_staging_database`
- `target_staging_database` or `staging_database`
- `staging_database_for_table`
- `target_database`

Data Mover supports using either the `staging_database` or `target_staging_database` tags to specify the target staging database, but not both at the same time.

When these values are specified, they are applied to all the table objects included in the job request. For non-table or view objects, if these values are supported, they need to be specified at the object level. These tags are optional, and must be positioned after the `<force_utility>` tag.

```
<overwrite_existing_objects>true</overwrite_existing_objects>
<force_utility>tptapi</force_utility>
<source_staging_database><name>stagingSourceDB</name></source_staging_database>
<target_staging_database><name>stagingTargetDB</name></target_staging_database>
<staging_database_for_table><name>stagingDBForTable</name></staging_database_for_table>
<target_database><name>targetDB</name></target_database>
...
```

System Level Specification Using the `save_configuration` Command

You can also specify values for staging and target database tags at the system level. Specify at the system level by executing the `save_configuration` command, the same command used to set other Data Mover configuration properties.

When `staging_database`, `staging_database_for_table`, `source_staging_database`, `target_staging_database`, and `target_database` tag values are specified at the system level, Data Mover applies them to all table objects included in the job request.

A property named `system.default.database.enabled` controls the system level default values for the target and staging databases. For each target system, you can set three values for the default databases. Within the `systemLevelDatabase` element, the `system` tag is mandatory and the databases tags are all optional. The following is an example of the new property in the `save_configuration` XML file:

```
<property>
  <key>system.default.database.enabled</key>
```

```

    <value>true</value>
    <defaultDatabases>
        <systemLevelDatabase>
            <system>targetSystem1</system> <!-- Tdpid here -->
            <source_staging_database>stagingDB1</
source_staging_database>
            <!--target_staging_database>stagingDB1</
target_staging_database-->
            <staging_database>stagingDB1</staging_database>
            <staging_database_for_table>stagingDBForTable1</
staging_database_for_table>
            <target_database>targetDB1</target_database>
        </systemLevelDatabase>
        <systemLevelDatabase>
            <system>targetSystem2</system>
            <!-- only specify the target database for targetSystem2
-->
            <target_database>targetDB2</target_database>
        </systemLevelDatabase>
    </defaultDatabases>
    <description>Purpose: Enable/Disable the default target/staging
databases at the system level. Default value false means disabled.</
description>
</property>

```

Priority of Values Specified for Target and Staging Databases

If you specify the staging and target database values at more than one level, it is important to know what the priority is. The general rule is that the smallest scope has the highest priority. The following table shows the priority levels:

Staging or Target Databases	Priority
Object level	Highest
Database Level	Higher
Job Level	High
System Default Level	Low

Dynamic Evaluation of the System-Level Default Databases

The system level default database values are not part of the job definition. They are retrieved from the configuration at job creation or starting time. When a job is executed by the start or move command, the default database values are re-evaluated dynamically to replace the existing system default values. Because of this, the same job definition can have different target or staging databases when executed, depending on the run time system level default databases. For a job executed by the restart command, the system level default database values are not re-evaluated; the database values set in the prior execution are preserved.

About Copying Objects

You can use the Data Mover command-line interface to copy individual database objects, as well as to copy entire databases, including all of the objects they contain.

When you create a job definition, in addition to specifying basic source and target database information such as database names, users and passwords, you can specify exactly which objects to copy by creating or editing the XML file that defines the job.

The Object List

The object list is part of the Data Mover XML schema and specifies the database objects to be copied.

The following example shows the object list portion of an XML file:

```
<database selection="unselected">
  <name>dmguest</name>
  <table selection="included">
    <name>table1</name>
  </table>
  <table selection="unselected">
    <name>table2</name>
  </table>
  <table selection="included">
    <name>table3</name>
  </table>
</database>
```

Selection parameter values determine which objects are copied, as described in the following table.

Value	Action Taken
included	The specified object is copied.
unselected	The specified object is not copied.

Objects Supported During Moves Between Databases

When selecting object types for copying, you can select a subset of the viewable objects.

Components Available	Teradata to Teradata
View Objects	<ul style="list-style-type: none"> • Database • Users • Tables • Most Object Types ¹
Select Objects	<ul style="list-style-type: none"> • Database • Users • Tables • Most Object Types ¹
Relocate to Target	Database
Table Settings	Yes
Column Settings	No
Foreign Server Objects	Yes
Function Alias Objects	Yes
¹ Most Teradata Database object types are viewable and selectable.	

Naming Objects

Databases and tables specified for the command line must follow Teradata naming rules. Names can contain only the following characters:

- a - z
- A - Z
- _
- \$
- #
- 0 - 9
- (blank space)

Command line or job XML expects the database, table, and column object names to be identical to the database object names. If the object starts with a special character or contains any leading or trailing spaces, do not add double quotes. If the object name contains double quotes, do not add extra double quotes to escape it. Teradata systems support leading spaces only, not trailing spaces. For more information, see system documentation.

The following table provides examples of object names.

Database Object Name	Command Line Name
[DBC]	[DBC]
[Ide#..2]	[Ide#..2]
["SPLCh"Table]	["SPLCh"Table]
[SPLCh Table]	[SPLCh Table]
[21SPL]	[21SPL]

The Move Command

The move command creates a job to copy specified database objects from one database to another, then starts the job immediately. Use the move command to create and start a job quickly without an XML object list by specifying tables on the command line.

Note:

For jobs that use Teradata DSA, the database user for the target system to which you are copying tables must have RESTORE privileges and must not be logged on while the job is running. This causes a failed job.

Teradata Database MAPS Architecture Feature Support

Data Mover supports the Teradata Database MAPS Architecture feature (MAPS) **map** and **colocate** parameters when copying the following objects:

- Table
- Index (hash or join)
- Foreign server – data transfers using the following utilities:
 - JDBC
 - TPTAPI
 - T2T

The following table describes where you can specify the **map** and **colocate** parameters. They are listed in the order of parameter priority:

Level	Description
Object	Uses the map and colocate parameters for the specified object (table, index, or foreign server).
Database	Uses the map and colocate parameters for all specified objects within the database.
Job	Uses the map and colocate parameters for all objects in the specified job.

Level	Description
System	Uses the system-level map for all jobs, which uses the specified system as the target system.

Rules and Restrictions

- The **map** and **colocate** parameters are not supported in databases prior Teradata Database 16.10.
- The target user must have permissions on target maps.
- The target default map is the **help session** map when the object owner and target login user are the same, or the map is null for the target owner. When the target object owner and the user are different, the system uses the map of the target owner as the target default map. The **help session** is a query which displays information on the current session, see the Teradata Database Tools and Utilities guide.
- Creating a **colocate** name is only allowed for **sparse** maps and can be any valid name, the default name is DatabaseName_TableName.
- DSA supports user-defined **map** values when copying tables, but does not support user-defined **colocate** values. When a map is not specified, DSA tries to find a map on the target system with a similar definition as the source table. If no map is found or user does not have permissions for sparse maps, DSA uses the default map.
- When DDL is enabled, make sure the following map and colocate parameters match:
 - If map and colocate parameters exist on the source and target systems, they should match.
 - If map and colocate parameters are provided in a job, then they should match the map and colocate parameters on the target system.

XML Examples

Specify system level map parameters through configuration.xml and save using the save_configuration command:

```
<property>
  <key>map</key>
  <value>>false</value>
  <maps>
    <systemLevelMap>
      <system>prodSystem1</system>
      <map>sparsemap4amp</map>
    </systemLevelMap>
  </maps>
</property>
```

Specify job level **map** and **colocate** parameters for create or edit jobs:

```
<dmCreate>
  . . . . .
```

```

    <log_to_even_table><log_to_event_table>
    <map>TD_SparseMap</map>
    <colocate>ColocationName2</colocate>
    . . . . .
</dmCreate>

```

Specify database level **map** and **colocate** parameters as shown in the following example:

```

<database selection="unselected">
    . . . . .
    <compare_dll></compare_dll>
    <map>TD_SparseMap</map>
    <colocate>ColocationName2</colocate>
    . . . . .
</database>

```

To specify table, index (hash and join) and foreign server level **map** and **colocate** parameters, refer to their respective sections in this guide.

About Copying Tables

You can copy an individual table using the Data Mover command-line interface by specifying the value `table selection="included"` and the name of the table. In the XML example, `table1` and `table3` are copied as they specify the value `table selection="included"`. `Table2` in the objects list is not copied because the value is set to `table selection="unselected"`.

```

<database selection="unselected">
  <name>dmguest</name>
  <table selection="included">
    <name>table1</name>
  </table>
  <table selection="unselected">
    <name>table2</name>
  </table>
  <table selection="included">
    <name>table3</name>
  </table>
</database>

```

Usage Notes

- If the object being copied does not exist on the target database, it is created on the target database. If the object does exist on the target database, the object is overwritten – unless the `overwrite_existing_objects` property is set to `false`. In that case, a create time error is generated.
- Tables can be copied between systems supported by Data Mover.

Verifying Row Counts

You can verify row count for tables when moving them from Teradata to Teradata.

Note:

Values are not case sensitive.

1. To validate row counts for a table, add `validate_row_count` to the table object.

```
<table selection="included">
  <name>MyTestTable_1</name>
  <validate_row_count>all</validate_row_count>
</table>
```

The following table lists valid values for `validate_row_count`:

Value	Description
ALL	Row count check is performed on the entire table. This option is available for partial table copies. For example, if you want to validate the source and target tables are synchronized, check the row count on the full table after the partial table has been copied.
NONE	No row count check is performed. This is the default.
PARTIAL	Row count check is performed on subset of data that was copied. This option is available only for partial table copies. If used for full table copies, an error results.

For partial row count validation, a WHERE clause is provided for both the source and target systems. In most cases, Data Mover does not replace the table name in the WHERE clause when performing a row count validation query. There is one exception: if the WHERE clause contains a fully qualified source table name and the table is relocated or renamed, the fully qualified source table name is replaced with the fully qualified target table name.

Examples of the expected behavior are the following:

Source Object	WHERE Clause	Target WHERE Clause
Fully qualified source object	WHERE col1 in select c1 from sourcedb.sourcetable;	WHERE col1 in select c1 from targetdb.targettable;
	WHERE col1 in select c1 from "sourcedb"."sourcetable";	WHERE col1 in select c1 from "targetdb"."targettable";

Source Object	WHERE Clause	Target WHERE Clause
Non-fully qualified source object	WHERE col1 in select c1 from sourcetable;	WHERE col1 in select c1 from sourcetable;
Fully qualified non-source object	WHERE col1 in select c1 from sourcedb.tableA;	WHERE col1 in select c1 from sourcedb.tableA;

- View the row count validation results by typing the `status` command with an *output_level* of 3 or more. This displays the table name, row count on the source and target systems, whether the row count is synchronized, and if the validation type was PARTIAL or ALL.

About Overriding Locks for Access

The Teradata Database system places a lock on a table during every table transaction. For example, if a transaction is taking place on the source table, you cannot run a read operation until the previous transaction is complete. However, the override lock for access feature allows you to override the default lock and read data from the source table, which could be in the middle of a transaction. This functionality (also known as a *dirty read*) contains the risk of copying data to a target table that has not been committed on the source table:



NOTICE

Be extremely careful when using this feature.

This feature is optional and can only be specified using XML, not the command-line interface. In the object list XML file, override the default lock by adding an `override_lock_access` element that is set to `true`. The functionality works on a per-table basis, therefore specify the `override_lock_access` element under the specific table where the override can occur.

The following example shows the correct location for the `override_lock_access` element.

```
<database selection="unselected">
  <name>srcDatabase</name>
  <table selection="included">
    <name>srcTable</name>
    <validate_row_count>all</validate_row_count>
    <override_lock_access>true</override_lock_access>
    <compare_ddl>true</compare_ddl>
  </table>
</database>
```

The default value for the `override_lock_access` element is `false`.

Note:

Only Teradata PT API, T2T, and JDBC utilities support this feature. Teradata DSA cannot be used.

Restrictions on Overriding Locks for Access

- Functionality is restricted to tables only.
- The feature does not work if `force_utility` is set to Teradata DSA.
- If `force_utility` is not specified and Data Mover selects Teradata DSA to copy tables, the Override Lock for Access feature is not used.

For more information, see:

- *Teradata® Database SQL Request and Transaction Processing*, B035-1142
- *Teradata® Database SQL Data Manipulation Language*, B035-1146

About Using Export Without Spool

You can preempt spooling preceding a table copy when using Teradata PT API. The Teradata Database writes data to temporary disk space (*spool space*) where it is held until the select is complete. This action is known as *spooling*. Although spooling occurs by default to prevent data inconsistencies, it also adds to the time required to copy database objects. To improve performance for a specific table copy in jobs that use Teradata PT API, disable spooling for the table.

Disadvantages of using this feature when copying tables with the Teradata PT API operator include the following:

- Locks are maintained during the entire copy process.
- Because the spooling phase detects data conversion errors, eliminating spooling when copying a table can introduce conversion errors. If this happens, correct the errors and restart the job.

The `export_without_spool` element is optional and enabled by default. This element can only be specified in the XML, but not as a parameter on the command-line interface. The element is only available for Teradata PT API jobs, and only applies to Teradata Database 13.10 and later. If you specify `true` for the value of the `export_without_spool` element in the XML, and the source or target is earlier than Teradata Database 13.10, or if the job uses a different utility, the element is ignored. This element is under the `table` element in the XML file. The following table lists valid values for the `export_without_spool` element.

Value	Description
false	Data Mover writes to spool space when copying a table.
true	Data Mover copies a table without writing to spool space. This is the default value.

Value	Description
unspecified	Equivalent to omitting the <code>export_without_spool</code> element in the XML. Data Mover does not use spooling when exporting data from the source Teradata Database in a table copy job that uses the Teradata PT API.

Because you can only disable spooling for a table copy, specify the value `true` for the `export_without_spool` element under the `table` tag. In the following example, table `srcDatabase.Employees` is copied, but spooling is disabled for the table.

```
<database selection="unselected">
  <name>srcDatabase</name>
  <table selection="included">
    <name>Employees</name>
    <export_without_spool>true</export_without_spool>
  </table>
</database>
```

About Partial Copies

By default, Data Mover uses Teradata PT to perform partial copies on tables and views. The `WHERE` clause provided in the job definition is appended to the `SELECT` statement used by the Teradata PT export operator when exporting data from the source system. The Teradata PT load, update, or stream operator is used to insert the partial data into the target system.

If the target table already exists and is not empty, Data Mover creates a target staging table and places the data there first. After the partial data is loaded on to the target staging table, Data Mover uses the key columns provided in the job definition to determine which rows are being updated and which rows are new and can be inserted. The target staging table is dropped once all of the data is in the target table.

You can use Teradata QueryGrid to perform partial copies on tables or views by specifying a foreign server in the job definition. The partial copy performs in the same manner as Teradata PT, except with Teradata QueryGrid being used to copy the partial data from the source system to the target system.

You can use DSA to perform partial copies for tables and views by telling Data Mover to use a source staging table. When a source staging table is used, Data Mover first creates a source staging table, then copies the partial data into that table. Then, DSA copies the full source staging table to the target system. If the target table already exists and is not empty, the source staging table is copied into a target staging table where the data can be transferred over into the target table using the same methods as Teradata PT.

About Performing Partial Copies

When performing a partial copy using XML, embed a `sql_where_clause` tag and one or more `key_column` tags within the `table` tag of the table to perform the partial copy. The `sql_where_clause` tag specifies the SQL `WHERE` clause used for the extract side of the partial copy job. The `key_column`

tag is used to specify the name of the columns that uniquely identify each row when upserting them into the target table.

For example:

```
<table selection="included">
  <name>Table1</name>
  <sql_where_clause><![CDATA[ WHERE col1 > 4]]></sql_where_clause>
  <key_columns>
    <key_column>col1</key_column>
    <key_column>col2</key_column>
  </key_columns>
</table>
```

Inside the `sql_where_clause` tag, the WHERE clause must be surrounded by the special CDATA tag, `![CDATA[where clause]]`, and the WHERE clause must include the WHERE keyword. Do not use a ; character at the end of the SQL.

When performing a partial copy using Teradata PT, the Teradata PT export operator does not support WHERE clauses that can be satisfied by a single AMP operation. For example, a SELECT statement with a constraint containing an equality condition on the primary index or unique secondary index columns of a table such as, `WHERE col1 = 1` where `col1` is the primary key for the table). In these cases, you should use an alternative copy method.

Note:

For optimal results, if at least one of the primary indexes is unique, specify all primary indexes as key columns. For PPI tables, specify all primary indexes and the partitioned columns as key columns. Omitting the partitioned columns when specifying primary indexes can result in job failure.

About Performing Partial Copies using Source Staging Table and DSA

When performing a partial copy of a view or table using a source staging table, add the `use_source_staging_table` tag to the XML and set the tag to true. This enables Data Mover to use DSA when copying the table or view when DSA is a valid copy method.

```
<table selection="included">
  <name>f_table1</name>
  <use_source_staging_table>TRUE</use_source_staging_table>
</table>
```

For information on specifying a source staging database, see [Specifying Source Staging at the Database and Table Levels](#).

About Using staging_to_target Element in XML

Data Mover uses target staging tables when performing partial copies or full table copies where the target table already exists. The value of the `staging_to_target` element specifies how to copy data from a target staging table to a target table, overriding the method Data Mover normally uses to copy data from staging to the target table. The Data Mover method uses MERGE or DELETE together with INSERT/SELECT, depending on factors such as key columns, database version, and so forth.

Note:

The override fails if you have specified the MERGE value for the `staging_to_target` element. Data Mover determines that the MERGE method is not successful in copying data, resulting in an error.

Use the `staging_to_target` element if you do not have UPDATE permission required by the MERGE statement. However, use the parameter carefully as Data Mover automatically chooses the most efficient way of copying data.

The `staging_to_target` element is optional and can only be specified in the XML file, not as a parameter to the command-line interface.

The `staging_to_target` element is under the `key_columns` element in the XML file. The following table lists the valid values for the `staging_to_target` element.

Value	Description
NOT_SPECIFIED	(Default) Recommended because it specifies that Data Mover choose the most efficient way to copy data from the target staging table to the target table.
MERGE	Specifies the use of the MERGE statement to copy data. Data Mover verifies whether the MERGE statement can be used to copy data. An error results if the MERGE statement cannot be used. Note that the MERGE statement cannot be used to copy multiset tables.
DELETE_INSERT	Specifies rows from the target table are deleted with the DELETE statement, then copied from the staging table to the target table with the INSERT/SELECT statement. Rows to be deleted depend on the job. With a partial table copy, only the rows that match the SQL query are deleted. With a full table copy, all rows are deleted.
INSERT_ONLY	Specifies no rows are deleted from the target table. Instead, the INSERT/SELECT statement copies rows from the staging table to the target table.
DELETE_DISTINCT_INSERT	Specifies rows from the target table are deleted with the DELETE DISTINCT SELECT statement, then copied from the staging table to the target table with the INSERT/SELECT statement. This method is used by partial copy only, and only when explicitly specified. This method improves delete performance when key columns are non-unique keys.

Because functionality of the `staging_to_target` element is valid for individual tables, specify the table to which the `staging_to_target` feature applies, as in the following example.

```

<database selection="unselected">
  <name>srcDatabase</name>
  <table selection="included">
    <name>srcTable</name>
    <sql_where_clause>
      <![CDATA[WHERE colA > 100]]>
    </sql_where_clause>
    <key_columns>
      <key_column>colA</key_column>
    </key_columns>
    <staging_to_target>insert_only</staging_to_target>
  </table>
</database>

```

About Force Target Staging Table

Data Mover normally creates staging tables when the target table exists and is not empty during a copy, except when copying a full or partial table in the following circumstances:

- Using DSA to copy and the target table exists
- Using TPT, T2T, or JDBC to copy and the target table exists and is empty

When you want Data Mover to create a staging table on the target database for the mentioned scenarios, set `force_target_staging_table` to true. A staging table is not created when the target table does not already exist on the target system, regardless if `force_target_staging_table` is set to true.

```

<table selection="included">
  <name>Table1</name>
  <force_target_staging_table>true</force_target_staging_table>
</table>

```

About Copying Tables with Referential Integrity

In most cases, you can copy tables that have referential integrity from the source to the target using Teradata PT API, Teradata JDBC, and Teradata DSA.

A table has *referential integrity* if any of its foreign key (FK) columns reference primary key (PK) columns in another table. Referential integrity helps make sure data integrity and consistency between PK and FK columns.

Referential integrity constraints can be added to a table when using CREATE TABLE or ALTER TABLE statements. In the Teradata Database, you can declare standard, batch, and soft referential integrity. In the following examples of the different types of referential integrity, the composite foreign key column-set (Name, Deptid) references the primary key columns (Name, Deptid) in the parent table db.employee_PK:

Referential Integrity	Example	Description
<i>Standard</i>	FOREIGN KEY (Name , Deptid) REFERENCES db.employee_PK (Name , Deptid))	Enforces referential integrity at the row level.
<i>Batch</i>	FOREIGN KEY (Name , Deptid) REFERENCES WITH CHECK OPTION db.employee_PK (Name , Deptid))	Enforces referential integrity for an entire implicit transaction; useful when a single statement inserts multiple rows into a table.
<i>Soft</i>	FOREIGN KEY (Name , Deptid) REFERENCES WITH NO CHECK OPTION db.employee_PK (Name , Deptid))	Explicitly instructs the Teradata Database not to incur system overhead by enforcing referential integrity on the PK-FK relationship.

The following scenarios describe how Data Mover handles copying tables with foreign keys.

Source Table	Target Table	Data Mover Results
FK definition exists	Table does not exist.	Copies the source table to the target with the FK definition of the source.
FK definition exists	Has no FK definition.	Copies the source table to the target with no referential integrity, therefore the target table has no FK definition.
FK defined for col1 and col2	Has FK defined for col3 and col4.	Copies the source table to the target but the target table keeps its original FK definition (FK on col3 and col4). The target table does not use the FK definition of the source table (FK on col1 and col2).

The following restrictions apply to copying tables that have referential integrity:

- If a parent table is copied to the target, the table cannot exist on the target with a child that is not copied as part of the job for either of the following two conditions:

Condition	Example	Error Information
Child has hard referential integrity to the table.	Source tables A and B, where A is the parent of B and C.	An error occurs when copying source tables A and B where C has hard references to A on the target, but C is not being copied. The job should be created when only moving A and C, because B is soft referential integrity to A.
	Target tables A, B, and C, where A is the parent of B and C; B has soft referential integrity reference to A; and C has hard referential integrity references to A.	

- When copying a child table, the parent table can only be renamed or relocated if the child table already exists on the target database, or if the parent table exists in the same database on the target as it does in the source.
- If the table does not exist on the target, the source table definition is used to create the table on the target. Because the Teradata Database represents the primary key as a unique index in a table definition, the target table is created with a unique index instead of a primary key. Child tables can reference tables with a unique primary or secondary index.
- If a job copies the entire database with Teradata DSA, and the database has tables with foreign keys, the target tables are created without foreign keys.
- If the source and target tables have different foreign keys, the foreign keys on the target table are maintained after the table copy.
- A child table can only be copied without copying its parent table only if the child table already exists on the target database, or if the parent table exists in the same database on the target as it does in the source.
- When moving tables, if a parent table has a GENERATE ALWAYS identity key and that key is used as a foreign key for a child table, the tables can be moved only with Teradata DSA. Partial copies are not supported for this type of RI table with DSA, TPT, or JDBC.

Two tables are said to have *circular references* if a primary key in the parent table (PT1) references a foreign key in the child table (CT1) and a primary key in CT1 references a foreign key in PT1. That is, a mutual PK-FK relationship exists between both tables. When copying source tables with circular references, Data Mover can only copy to the target if the target tables have no mutual PF-FK relationship; there must not be any circular references between the target tables. An error occurs if tables do not exist on the target when Data Mover tries to copy source tables with circular references.

About Copying Multi-Set Tables

Data Mover includes an override option that enables advanced users to force the use of the TPTAPI_LOAD operator for multi-set tables. This is accomplished by setting the allowTptLoadForMultiset table attribute to true for multi-set tables when using force_utility TPTAPI_LOAD. The table is moved to the target using TPTAPI_LOAD; duplicate rows may be lost if the table has PI and does not exist on the target system.

Note:

When the allowTptLoadForMultiset table attribute is set to true, do not specify validate_row_count, or the job may fail during validation.

NOTICE

This feature should be used by only advanced users who are aware that duplicate records may be lost when the table is moved to the target.

About Copying Temporal Tables

In a conventional database system, data stored in tables is considered valid at the present time. Because conventional tables are limited to a current snapshot of reality, they do not retain data that was present in the past or that will be valid in the future. A *temporal* database allows you to store data related to points in time by providing temporal data types and storing information related to the past, present, and future. For example, a table in a temporal database system can store data about inventory levels at the end of each month of the year. By storing the temporal dimensions of data in tables, you can query historical and prospective data trends.

A *temporal table* contains one or both of the following:

Time	Description
ValidTime	Data modeled on the real world using a time period during which a fact (a row) is true (valid) in the real world. A value for the DATE or TIMESTAMP column in the table defines the time that a fact is known to be valid.
TransactionTime	Data using a time period beginning at the moment a fact (a row) is recorded (committed) in the database and ending when the fact is superseded through an update, rollback, or deletion. A value for the TIMESTAMP column in the table defines the transaction time of the fact.

You can copy a temporal table, associated join and hash indexes, and triggers to the target system. To copy temporal tables from source to target, you must have NONTEMPORAL privileges on the target database. In the following XML example, the temporal table in Employee_BT_1 is copied to the target system:

```
<table selection = "included">
  <name>Employee_BT_1</name>
  <target_database>
    <name>MyDatabase</name>
  </target_database>
</table>
```

If a `force_utility` is not specified, Data Mover automatically selects Teradata DSA when copying temporal tables. If DSA is not available, the Teradata PT update operator is selected. If you force the use of Teradata PT, the Teradata PT update operator is chosen as the copy method. When copying entire databases with DSA, all temporal tables on the source are copied to the target. When copying a temporal table, you can rename or relocate temporal tables on the target system or perform partial copies of the temporal tables.

When copying temporal tables, non-sequenced queries select the source rows that need to be copied to the target. For example:

```
<NONSEQUENCED VALIDTIME> <AND> <NONSEQUENCED TRANSACTIONTIME> SELECT *
FROM Source_Table;
```

Non-sequenced queries do not place any special semantics on temporal columns, treating them as any other regular columns. Therefore all current, history, and future rows that are open or closed are copied to the target.

The primary key constraint of a temporal table is maintained through a system-defined join index. This system-defined join index is automatically generated by the Teradata Database when the table is created. When copying a temporal table, you do not need to separately copy the system-defined join index. DSA copies the system-defined join index setting. For other copy methods, the system-defined join index already exists on the target if the table already exists or it is automatically generated when the table is created as part of the job.

The following scenarios result in an error when copying temporal tables:

- Copying using the TPT API LOAD operator when the `allowTPtLoadForMultiset` attribute is not set to true.
- Copying the temporal table and its system-defined join index as as a separate item rather than copying only the temporal table.

About Copying Tables using the MAPS Parameters

Using the **map** and **colocate** parameters are optional. The object-level parameters have a higher priority than database, job, and system-level parameters that use the MAPS feature.

The following example shows table objects being copied using the **map** and **colocate** parameters:

```
<database selection="unselected">
  <name>srcDatabase</name>
    <table selection="include">
      <name>srcTable</name>
      . . . .
      <compare_ddl><compare_ddl>
      <map>TD_SparseMap</map>
      <colocate>ColocationName2</colocate>
      . . . .
    </table>
  </database>
```

About Copying Foreign Tables

You can use the foreign tables option in the database to access native object stores. Data Mover can copy the definitions of a foreign table from one database to another using the same syntax as used when

copying a normal table object. Foreign table definitions can be copied as a single table or be included in an entire database copy.

Considerations when copying foreign tables are as follows:

- Only foreign table definitions can be copied
- Foreign tables can be copied along with regular tables
- Foreign table definition copy occurs at the post data move step
- Data is not copied with the foreign tables option, therefore, there are no utility requirements
- Only a few table-level properties are supported when copying foreign tables:
 - Rename
 - Relocate
 - MAPs
 - compare_ddl
 - copyStats

Any properties not listed here are not supported when copying foreign tables, such as staging database, row count validation, and so forth.

```
<database selection="unselected">
  <name>testDB</name>
  <table selection="included">
    <name>foreignTableName</name>
  </table>
</database>
```

About Copying Columns

You can copy a column using the Data Mover command-line interface by specifying the value `table_columns` when a table is selected.

```
<database selection="unselected">
  <name>dmguest</name>
  <table selection="included">
    <name>table1</name>
    <target_name>NewTable1</target_name>
    <table_columns>
      <name>column1</name>
      <target_name>targetColumn1</target_name>
      <type>varchar</type>
      <target_type>string</target_type>
    </table_columns>
  </table>
</database>
```

About Copying Join and Hash Indexes

Copying join and hash indexes between database versions has certain restrictions:

To copy a join or hash index from a source database to a target database, specify the `selection="included"` attribute of the `index` element when defining the join or hash index, as shown in the following example:

```
<indices>
  <index selection= "included">
    <name>Orders_HI</name>
    <index_database>west1000</index_database>
    <map>sparsemap4amps1</map>
    <colocate>sparsecol2</colocate>
    <index_type>HASH_INDEX</index_type>
  </index>
</indices>
```

The parameters and possible values for join and hash indexes are described in the following table:

Parameter	Description
colocate	Co-location name, for sparse map only.
index_database	Database name in which the join or hash index resides.
index_type	Type of index. Valid values are: <ul style="list-style-type: none"> • HASH_INDEX • JOIN_INDEX
map	Object map name.
name	Join or hash index name.
selection	Element that indicates whether the join or hash index is copied. Valid values are: <ul style="list-style-type: none"> • included to copy the join or hash index • unselected to exclude the join or hash index from being copied

Note:

The MAPS feature parameters, **map** and **colocate**, are supported only in target systems using Teradata Database 16.10 or later.

If you are copying a join or hash index along with the associated tables and the associated tables are being renamed or relocated, the join or hash index is created on the renamed or relocated table on the target.

The following code example creates a compressed multi-table join index, `Cust_Ord_JI`.

```
CREATE JOIN INDEX west1000.Cust_Ord_JI ,NO FALLBACK ,CHECKSUM = DEFAULT AS
SELECT (west1000.C.c_custid ,west1000.C.c_lname ),(west1000.O.o_orderid ,
```

```
west1000.O.o_orderstatus ,west1000.O.o_orderdate ) FROM
west1000.Customer C INNER JOIN west1000.Orders O ON west1000.C.c_custid
= west1000.O.o_custid )
PRIMARY INDEX ( c_custid );
```

The following code example creates a non-compressed multi-table join index, Cust_Ord_J2.

```
CREATE JOIN INDEX west1000.Cust_Ord_JI2 ,NO FALLBACK ,CHECKSUM = DEFAULT AS
SELECT west1000.C.c_custid ,west1000.C.c_lname,west1000.O.o_orderid ,
west1000.O.o_orderstatus ,west1000.O.o_orderdate FROM
west1000.Customer C INNER JOIN west1000.Orders O ON west1000.C.c_custid
= west1000.O.o_custid )
PRIMARY INDEX ( c_custid );
```

The following code example creates an aggregate join index, Monthly_Sales_JI.

```
CREATE JOIN INDEX west1000.Monthly_Sales_JI ,NO FALLBACK ,CHECKSUM = DEFAULT AS
SELECT COUNT( * ) (FLOAT, NAMED CountStar ),west1000.Daily_Sales.item_id
(NAMED Item ),
EXTRACT(YEAR FROM (west1000.Daily_Sales.sales_date ))(NAMED Yr ),
EXTRACT(MONTH FROM (west1000.Daily_Sales.sales_date ))(NAMED Mon ),
SUM(west1000.Daily_Sales.sales )(FLOAT, NAMED Sum_of_Sales )
FROM west1000.Daily_Sales GROUP BY west1000.Daily_Sales.item_id (NAMED Item ),
EXTRACT(YEAR FROM (west1000.Daily_Sales.sales_date ))(NAMED Yr ),
EXTRACT(MONTH FROM (west1000.Daily_Sales.sales_date ))(NAMED Mon )
PRIMARY INDEX ( Item );
```

The code example here creates a hash index, Orders_HI.

```
CREATE HASH INDEX west1000.Orders_HI ,NO FALLBACK ,CHECKSUM = DEFAULT
(o_custid ,o_totalprice ,o_orderdate )
ON west1000.Orders ;
```

When copying join and hash indexes, take the following into consideration:

- If the object being copied does not exist on the target database, it is created on the target database. If the object does exist on the target database, the object is overwritten – unless the `overwrite_existing_objects` property is set to false. In that case, a create time error is generated.
- Copying a table that exists on the target with an associated join or hash index overwrites the target table and recreates the join or hash index on the newly copied table.

Renaming and Relocating Hash and Join Indexes

Use the `target_name` and `target_index_database` elements to rename and relocate hash and join indexes.

Both elements are optional and can be used in any combination. If only the `target_name` element is specified, the target index database name will be the same as the source index database name. If only the `target_index_database` tag is specified, the target index name will be the same as the source index name.

1. Add the `target_name` and `target_index_database` elements under the `index` element.

This is shown in the following example:

```
<indices>
  <index selection="included">
    <name>SrcHashIndex</name>
    <target_name>TargetHashIndex</target_name>
    <index_database>SrcDatabase</index_database>
    <target_index_database>
      <name>TargetDatabase</name>
    </target_index_database>
    <index_type>HASH_INDEX</index_type>
  </index>
</indices>
```

Specifying a Target Hash Index or Join Index

1. Add a `target_name` element.
2. Specify the name of the target index.

In the following example, `newindexname` is the name of the target index.

```
<target_name>newindexname</target_name>
```

Specifying a Target Index Database

1. Add a `target_index_database` element along with a child name element.
2. Specify the name of the target index database.

In the following example, `newdatabasename` is the name of the target index database.

```
<target_index_database>
  <name>newdatabasename</name>
</target_index_database>
```

About Copying Triggers

When copying a trigger from a source database to a target database, specify the selection="included" attribute of the trigger tag when defining the trigger, as shown in the code example here.

```
<triggers>
  <trigger selection= "included">
    <database>west1000</database>
    <subject_table_database>west1000</subject_table_database>
    <table>employee</table>
    <name>RaiseTrig</name>
    <action_time enabled="NO">BEFORE</action_time>
  </trigger>
</triggers>
```

Parameters for the Trigger Element

Parameter	Values
selection	<ul style="list-style-type: none"> Specify "included" to copy the trigger. Specify "unselected" so that the trigger is not copied.
database	Database name in which the trigger resides.
subject_table_database	Database name in which the table that is associated with the trigger resides.
table	Name of the table that is associated with the trigger.
name	Name of the trigger.
action_time	<ul style="list-style-type: none"> Specify "BEFORE" to place the trigger on the target table before the table is loaded. This results in a trigger action for each row that is copied from the source. Specify "AFTER" to place the trigger on the target table after the table is loaded. This results in no trigger action for the rows that are being copied.
enabled	<ul style="list-style-type: none"> Specify "YES" to enable the trigger on the target. Specify "NO" to disable the trigger on the target. <p>The value must be "YES" to enable the action_time parameter. If the value is "NO", the action_time parameter is invalid.</p> <p>To verify if a trigger is enabled or disabled on the target, use the SELECT query from dbc.TriggersX and look at the EnabledFlag column.</p>

Rules and Restrictions

- Copying a trigger without copying its associated table results in an error.
- If the object being copied does not exist on the target database, it is created on the target database. If the object does exist on the target database, the object is overwritten – unless the `overwrite_existing_objects` property is set to false. In that case, a create time error is generated.
- When copying a table that already exists on the target with associated triggers, the target table is overwritten and the triggers are replaced on the newly copied table.
- Attempting to rename or relocate a table associated with or referenced by a copied trigger results in an error.
- If the trigger name in the trigger definition is not fully qualified, the trigger is created in the database having the same name as the source trigger database.
- If the table names in the trigger definition are not fully qualified, those tables must exist in the database having the same name as the source trigger database to avoid an error.

Example

This example creates a trigger on the source machine without fully qualifying object names in the definition:

```
CREATE TRIGGER RaiseTrig
  AFTER INSERT ON Employee
  FOR EACH ROW
  ( INSERT INTO SalaryLog VALUES ('Hello','Hi',23, 43); );
```

In this case, copying the trigger to the target creates the trigger in the database having the same name as the source trigger database. If the Employee and SalaryLog tables do not exist there, an error results.

Copying Triggers

1. Copy a trigger from a source database to a target database by specifying the `selection="included"` attribute of the trigger tag.

Note:

You cannot relocate or rename triggers.

Example:

```
<triggers>
<trigger selection= "included">
    <database>west1000</database>
    <subject_table_database>west1000</subject_table_database>
    <table>employee</table>
    <name>RaiseTrig</name>
    <action_time enabled="NO">BEFORE</action_time>
</trigger>
</triggers>
```

About Copying Functions

In Data Mover, use DSA to copy functions. Following restrictions apply:

- Grant the **create function** and **alter function** privileges to the source and target users.
- Copy only the following functions using DSA:
 - A - AGGREGATE_FUNCTION
 - F - STANDARD_FUNCTION
 - R - TABLE_FUNCTION
 - L - TABLE_OPERATOR
 - C - CONTRACT_FUNCTION
- Mention the specific name of the function as the function name. The specific name is always unique, but is not the function name.

Note:

The specific name of a function is defined in the SPECIFIC clause when created. If the SPECIFIC clause is not used, then database generates specific name for a function. You may find that value in the **tableName** field in **dbc.tablesV**. Refer to the *Teradata Vantage™ - SQL External Routine Programming*, B035-1147 to define a function.

- Use only DSA to copy functions. If you use other utilities, validation errors may occur.
-

Note:

The DLLs of the copied functions may not be in a usable state at the target database.

Example of the copying function:

```
<database selection="included">
  <name>DBName</name>
</database>
<functions>
  <function selection="included">
    <name>MY_Function</name>
    <database>MyDB</database>
  </function>
</functions>
```

About Copying Foreign Server Objects

Data Mover supports copying those foreign server objects definitions from one Teradata system to another Teradata system.

All the foreign server objects are kept in the database TD_SERVER_DB. This database exists in Teradata Database 15.0 or later.

For information about how to create, update, drop, or use foreign server objects, refer to the Teradata Database documentation.

Requirements and Restrictions

- Creating and copying foreign server objects requires that the respective foreign server packages are installed and configured properly on the Teradata Database system. If you need assistance, contact the Global Technical Support Center at <https://support.teradata.com>.
- The Data Mover job user must be granted permission for foreign server objects; otherwise, the job fails.
- Data Mover retrieves the foreign objects DDL by executing the `SHOW FOREIGN foreign_server_name` SQL statement.
- The Teradata Database system user must be granted permission for foreign server objects; otherwise, the job fails.
- Both source and target systems must be Teradata systems at version 15.0 or later.
- Data Mover does not parse a foreign server object DDL; the exact same DDL is run on the target system.
- Copy foreign server object definitions only.
- Relocation is not supported; all foreign server objects must be in database TD_SERVER_DB.
- Partial copy is not supported.
- Row count validation is not supported.
- Compare DDL is not supported.
- Using staging tables and staging databases is not supported when moving foreign server object definitions.
- Foreign server object definitions can be copied using DSA, TPT, or JDBC.
- If the object being copied does not exist on the target database, it is created on the target database. If the object does exist on the target database, the object is overwritten – unless the `overwrite_existing_objects` property is set to false. In that case, a create time error is generated.

Copying Foreign Server Object Definitions

The database that holds all the foreign server objects is TD_SERVER_DB.

1. Copy foreign server object definitions from a source database to a target database by copying the entire TD_SERVER_DB database or by copying foreign server objects only.

Copy Option	Description
Entire TD_SERVER_DB Database	All foreign server objects exist in database TD_SERVER_DB. You can copy the entire TD_SERVER_DB from one Teradata system to another Teradata system by specifying the database <code>selection="included"</code> attribute for database TD_SERVER_DB; no other attribute or properties are needed.

Copy Option	Description
	<pre> <dmCreate> ... <database selection="included"> <name>TD_SERVER_DB </name> </database> ... </dmCreate> </pre>
Only Foreign Server Objects Definition	<p>You can copy certain foreign server objects by specifying the <code>foreign_servers</code> section in the job definition, and setting the <code>selection="included"</code> attribute for each of the <code>foreign_server</code> sections. There is no need to specify the database name; they should exist in <code>TD_SERVER_DB</code> only. The <code>foreign_servers</code> section should be after the <code>views</code> section in the job definition.</p> <pre> <dmCreate> ... <foreign_servers> <foreign_server selection="included" > <name>MyServer</name> <map>sparsemap4amps1</map> <colocate>sparsecol2</colocate> </foreign_server> <foreign_server selection="included" > <name>MyServer2</name> <map>sparsemap4amps1</map> <colocate>sparsecol2</colocate> </foreign_server> </foreign_servers> ... </dmCreate> </pre>

Parameter	Description
colocate	Co-location name, for sparse map only.
map	Map name.
name	Foreign server name.

Note:

The MAPS feature parameters, **map** and **colocate**, are supported only in target systems using Teradata Database 16.10 or later.

About Copying Function Alias Objects

Function alias, also referred to as *function mapping*, allows you to create simple, alternate names to foreign function objects or foreign servers. A function alias is different from an SQL statement alias, where an alias

in a SQL statement is a temporary name, the function alias is a new permanent object in the database. A function alias hides details such as name and location for the foreign function being run.

Data Mover supports copying function alias objects from one Teradata system to another Teradata system that supports the function alias feature. Make sure all dependent objects in the function alias object definition exists in the target system before copying, Data Mover does not validate that all dependent objects in the function alias object exists in the target system.

The following parameters are used to copy a function alias object:

Parameter	Description
name	Function alias object name.
database	Owner or database name where the function alias object exists.

Requirements and Restrictions

- The Data Mover job user must be granted ACCESS permissions to function alias objects in the source system to run the SHOW FUNCTION MAPPING for DDL and DROP and CREATE function alias objects in the target system, otherwise the job fails at run time.
- All dependent objects in the function alias object must exist in the target system, otherwise the job fails at run time. Data Mover does not validate the existence of dependent objects.
- Both the source and target Teradata Database systems must be version 16.20 Feature Update 1 or later.
- Data Mover only supports copying function alias objects; renaming or relocating of objects to a different target database is not supported.
- Function alias objects can be copied using DSA, TPT, or JDBC.
- When copying an entire database using DSA, the function alias objects in the database are also copied.
- Data Mover does not parse the function alias object DDL, the same source DDL is run on the target system.

Example

The following example shows the function alias object DDL for a foreign server in the database TD_SERVER_DB. For more information about the function alias object feature, refer to the Teradata Database documentation.

```
CREATE FUNCTION MAPPING myDB.FAObjForForeignSer FOR
TD_SERVER_DB.td_conn_get_supp_version SERVER TD_SERVER_DB.MyForeignServerObject;
```

Copying Function Alias Object Definition

You can copy function alias objects from a source Teradata system to a target Teradata system.

1. Copy the full database using a utility such as DSA and specifying the selection="included" database attribute.

```
<dmCreate>
...
    <database selection="included">
        <name>MyDB</name>
    </database>
...
</dmCreate>
```

2. In the job definition, set the selection="include" attribute for each of the **function_alias** parameters.

```
<function_aliases>
    <function_alias selection="included" >
        <name>dmguest</name>
        <database>faObj</database>
    </function_alias>
    <function_alias selection="included" >
        <name>FAObjForForeignSer</name>
        <database>myDB</database>
    </function_alias>
</function_aliases>
```

About Copying Macros

A macro consists of one or more SQL statements that can be run by performing a single request. Each time the macro is executed, one or more rows of data are returned. For example, a macro could be created with following SQL statements:

```
CREATE MACRO Empinfo
AS (
    SELECT EmplId (TITLE 'Id')
    ,LastName      (TITLE 'Name')
    ,City          (TITLE 'City')
    FROM employee
    ORDER BY 2;
);
```

After creation, the macro definition can be retrieved by executing a show macro query on the macro.

```
SHOW MACRO Empinfo;
```

Copying Macros using Data Mover

Data Mover copies macros as follows:

- Data Mover retrieves the macro creation DDL by executing a `show macro dbname.macroname` SQL statement.
- Data Mover does not parse a create macro DDL; the exact same DDL is run on the target system. Data Mover does not relocate or rename macro objects.
- The underlying table for the macro must already exist on the target system or be copied with the job.
- If the macro name in the macro definition is not fully qualified, the macro is created in the database having the same name as the source macro database.
- If the table names in the macro definition are not fully qualified, those tables must exist in the database having the same name as the source macro database to avoid an error.
- If the object being copied does not exist on the target database, it is created on the target database. If the object does exist on the target database, the object is overwritten – unless the `overwrite_existing_objects` property is set to false. In that case, a create time error is generated.
- If a macro is copied during a full database copy, it is copied with the database using DSA. It is not copied separately.
- To copy a macro using DSA, see [About Copying Views, Macros, and Stored Procedures Using DSA](#).

Example: Job Request XML for Copying a Macro

An XML snippet for copying a macro is shown in the example here:

```
<dmCreate>
...
<macros>
  <macro selection="included">
    <name>macro1</name>
    <database>database1</database>
  </macro>
</macros>
...
</dmCreate>
```

Multiple macros can be specified within the `macros` element. Each macro element has two required elements: `name` and `database`.

Example: Job Request XML for Copying a Macro with its Underlying Table

The example here shows an XML snippet for copying a macro with its underlying table:

```

<dmCreate>
...
database selection="unselected">
  <name>database1</name>
  <table selection="included">
    <name>test</name>
    <validate_row_count>all</validate_row_count>
  </table>
</database>
<macros>
  <macro selection="included">
    <name>macro1</name>
    <database>database1</database>
  </macro>
</macros>
...
</dmCreate>

```

About Copying Datasets and Schemas

Data Mover supports copying objects associated with the Teradata Database DATASET type. The Teradata DATASET type is a complex data type (CDT) representing self-describing files that are interpreted based on a schema. DATASET data types where the underlying storage format is AVRO or CSV are supported. Moving AVRO and CSV schemas is also supported. A schema could be created with the following statements:

```

CREATE AVRO SCHEMA SYSUDTLIB.testSchema AS '{
  "namespace": "example.avro",
  "type": "record",
  "name": "User",
  "fields": [
    {"name": "name", "type": "string"},
    {"name": "favorite_number", "type": ["int", "null"]},
    {"name": "favorite_color", "type": ["string", "null"]}
  ]
}';

```

After creation, the schema definition can be retrieved by executing a show schema-type query on the schema.

```
SHOW AVRO SCHEMA testSchema
```


Copying Schemas Using Data Mover

Data Mover copies schemas as follows:

- Data Mover retrieves the schema creation DDL by executing a `show schema-type sysdtlib.schemaname` SQL statement.
- Data Mover does not parse a create schema DDL; the exact same DDL is run on the target system. Data Mover does not relocate or rename schema objects.
- If the object being copied does not exist on the target database, it is created on the target database. If the object does exist on the target database, the object is overwritten – unless the `overwrite_existing_objects` property is set to false. In that case, a create time error is generated.

Note:

Schemas are not overwritten if columns on the target system reference the schema.

- If copying a table with a DATASET column that references a schema, the schema must already exist on the target system or be copied with the job. Schemas are created in the SYSUDTLIB database.

Copying Datasets Using Data Mover

Query the column values to determine if the necessary information is provided to move the column as a DATASET column:

- The maximum length and DT type code are included in dataset type columns.
- The storage format column contains the following value when the column is a DATASET data type:

```
"Avro": DATASET stored as Avro
```

- A column entitled `datasetSchemaName` containing the name of the schema is associated with a particular column. If no schema is associated with a particular column, this field has a NULL value.

A dataset could be copied using the following command:

```
CREATE TABLE hcExampleTable(
    id INTEGER,
    avroFile DATASET(100) STORAGE FORMAT Avro WITH SCHEMA chemDatasetSchema
);
```

Example: Copying a Schema

The following example shows an XML snippet for copying a schema:

```
<dmCreate>
....
<schemas>
```

```

        <schema selection="included" >
            <name>chemDatasetSchema</name>
            <compare_ddl>false</compare_ddl>
        </schema>
    </schemas>
    ....
</dmCreate>

```

More than one schema can be specified within the schema element. Each schema element has one required element: name; and one optional element: compare_ddl.

Example: Copying a Schema That Does Not Exist on Target System

Tables that contain a dataset column that does not exist on the target system can be copied using an XML job request. The following example shows a job request XML snippet for a schema that does not exist on the target system:

```

<dmCreate>
    ....

    <database selection="unselected">
        <name>test1</name>
        <table selection="included">
            <name>myDatasetTable</name>
        </table>
    </database>

    <schemas>
        <schema selection="included" >
            <name>chemDatasetSchema</name>
            <compare_ddl>false</compare_ddl>
        </schema>
    </schemas>
    ....
</dmCreate>

```

About Copying SQL Stored Procedures

A SQL stored procedure is a set of SQL statements with an assigned name that is stored in the database in compiled form so that it can be shared by a number of programs. An example SQL stored procedure can be created with the following SQL statements:

```

CREATE PROCEDURE SP_deleteTestTable ()
BEGIN

```

```
DELETE FROM test;
END;
```

After the stored procedure is created, its definition can be retrieved by executing a `show procedure` query:

```
SHOW PROCEDURE SP_deleteTestTable;
```

Copying SQL Stored Procedures Using Data Mover

Data Mover copies SQL stored procedures as follows:

- Data Mover retrieves the SQL stored procedure creation DDL by executing a `show procedure dbname.spname` SQL statement.
- Data Mover does not parse a create procedure DDL; it executes the exact same DDL on the target system. It does not relocate or rename stored procedure objects.
- If the procedure name in the procedure definition is not fully qualified, the procedure is created in the database having the same name as the source procedure database.
- If the table names in the procedure definition are not fully qualified, those tables must exist in the database having the same name as the source procedure database to avoid an error.
- If the object being copied does not exist on the target database, it is created on the target database. If the object does exist on the target database, the object is overwritten – unless the `overwrite_existing_objects` property is set to false. In that case, a create time error is generated.
- If a procedure is part of a full database copy, the procedure is copied with the database using DSA. It is not copied separately.

When copying a SQL stored procedure, the underlying table for the procedure must already exist on the target system, or be copied with the job. Depending on the stored procedure creation SQL, a JDBC error may occur when creating a stored procedure without the underlying table.

If you want to copy a SQL stored procedure without copying the underlying table, you can use DSA. See [About Copying Views, Macros, and Stored Procedures Using DSA](#).

Example: Job Request XML for Copying a SQL Stored Procedure

An XML snippet for copying a SQL stored procedure is shown here:

```
<dmCreate>
...
<database selection="unselected">
  <name>database1</name>
  <table selection="included">
    <name>test</name>
  </table>
</database>
```

```

<stored_procedures>
  <stored_procedure selection="included">
    <name>macro1</name>
    <database>database1</database>
  </stored_procedure>
</stored_procedures>
...
</dmCreate>

```

Example: Job Request XML for Copying a SQL Stored Procedure with its Underlying Table

An XML snippet for copying a SQL stored procedure with its underlying table is shown here:

```

<dmCreate>
...
<database selection="unselected">
  <name>database1</name>
  <table selection="included">
    <name>test</name>
    <validate_row_count>all</validate_row_count>
  </table>
</database>
<stored_procedures>
  <stored_procedure selection="included">
    <name>sp_test</name>
    <database>database1</database>
  </stored_procedure>
</stored_procedures>
...
</dmCreate>

```

About Copying Statistics

Statistics provide information to Teradata Optimizer about the number of rows per value, and are used by the Teradata Database to evaluate and choose an optimal plan for accessing data. Using statistics improves the performance of complex queries and join. This is helpful in accessing a column or index with uneven value distribution. Statistics remain valid when a system is reconfigured.

Note:

Collected statistics are not automatically updated by the system. You are responsible for recollecting statistics to make sure Teradata Optimizer can make accurate decisions.

Requirements and Restrictions

A user must have SELECT access rights on dbc.IndexStats, dbc.ColumnStats, and dbc.MultiColumnStats views.

- When copying an entire database on which statistics have been collected, DSA copies the statistics even if the copyStats attribute is false or not specified. To avoid this, use Teradata PT API or Teradata JDBC to copy the database.
- If copying a database using DSA, any statistics associated with the table or indexes in that database are copied by default.
- When overwriting join or hash indexes on the target, statistics that were collected on the target indexes are dropped.
- The copyStats attribute is not displayed by default in the XML for tables, join indexes, and hash indexes that have had statistics collected. To copy statistics, enter the attribute in the XML manually.
- Any new statistics (column, multi-column, or index) added after job creation but before job is run, is copied as part of the job.
- Data Mover uses the output from SHOW STATISTICS to copy statistics to the target table.

Collecting and Copying Statistics

When copying statistics from the source to the target, first collect statistics on the source table, join index, or hash index. If statistics on the source have not been collected before trying to copy statistics from the source to the target, an error results.

1. Create a table with n columns.
2. Populate the table with rows.
3. Use SQL to collect statistics: `COLLECT STATISTICS <TableName> Column (Cols);`
 - To collect statistics about multiple columns, provide a list of columns.
 - To collect statistics about a named join or hash index, type `COLLECT STATISTICS <TableName> INDEX "IndexName"`.
 - To copy statistics after they have been collected, set the copyStats attribute to true. Below is an XML example of copying the statistics for a table, and the table itself, which is specified by the selection="included" attribute of the table element.

```
<database selection="unselected">
  <name>dbName</name>
  <table selection="included" copyStats="true">
    <name>MyTable</name>
```

```

        </table>
</database>

```

Statistics can be copied without copying the table itself by specifying the `selection="unselected"` attribute of the table element if the table already exists on the target. This rule also applies to hash and join indexes.

If statistics are copied with the object, the object does not have to exist on the target prior to the copy. If the object already exists on the target, the object is overwritten during the copy, and statistics are copied to the target.

Below is an XML example of copying statistics for a hash index:

```

<index selection="unselected" copyStats="true">
  <name>Cust_Ord_HI </name>
    <index_database>MyDB</index_database>
    <index_type>HASH_INDEX</index_type>
</index>

```

Note:

The `copyStats` attribute is valid only for tables, hash indexes, and join indexes.

Verifying Statistics

1. Compare the output of `HELP STATISTICS <Table> COLUMN <Col>` on the source and the target.

Specifying Copy Statistics at Job Level

Specifying `copyStats` at the job level allows for copying statistics in bulk instead of individually at the object level. When enabled, all tables and indexes (hash or join) in the job inherit the statistics at the job level. If a different value is specified at the table or index (hash or join) level, the job-level value is overridden.

1. In the job definition, set `copyStats` to `true` as in the following example:

```

<dmCreate>
...
<copyStats>true</copyStats>
<database selection="unselected">
  <name>database1</name>
  <table selection="included">
    <name>test</name>
    <validate_row_count>all</validate_row_count>
  </table>
</database>

```

```

    </table>
</database>
...
</dmCreate>

```

About Copying View Definition and View Data

This feature enables you to copy a view definition and the underlying view data from a source system to a target system.

When copying view data, TPTAPI_UPDATE, TPTAPI_STREAM, T2T, and JDBC can be used for a full view copy; DSA can be used for a partial copy when a source staging table is specified. A full view copy using DSA is allowed only when a source staging table is specified and the source view contains columns with the AVRO/CSV storage format. TPTAPI_LOAD cannot be used to copy view data because the target or staging table that Data Mover creates is a multiset table, which TPTAPI_LOAD does not support.

Restrictions When Copying Data from the Underlying View

The following restrictions apply when copying data from the underlying view:

- TPTAPI_LOAD cannot be used as the load utility when moving underlying data from the view.
- Do not use DSA as the load utility when moving underlying data from the view and a source staging table is not used.

Row Count Validation

You can validate the row counts of the source view and the target table after copying data from the underlying view. In order to perform row count validation, the `validate_row_count` element must be specified for the view.

```

<views>
  <view selection="included" copyData="true">
    <name>deptsals</name>
    <database>MyDb</database>
    <view_data_table>
      <target_table>TargetTable</target_table>
      <target_database>TargetDb</target_database>
    </view_data_table>

    <staging_database>
      <name>TargetDb_Staging</name>
    </staging_database>

    <validate_row_count>ALL</validate_row_count>
  </view>
</views>

```

When copying View data to a target view, the `validate_row_count` element validates the row counts of the source view and the target view.

Valid values for `validate_row_count` are ALL, PARTIAL, or NONE. The results of the row count validation can be seen in the status output. The row count validation is only performed when copying data from the view (`copyData="true"`). When copying only the view definition (`copyData="false"`), the row count validation is not performed even if the `validate_row_count` element is specified for the view.

About Copying a View Definition

The following xml demonstrates how to copy a view definition:

```
<views>
  <view selection="included">
    <name>deptsals</name>
    <database>MyDb</database>
  </view>
</views>
```

Set the selection as "included" to copy the view definition `deptsals` from the source system to the target system using the name of the view and the database where the view exists on the source system.

The following restrictions apply:

- If the object being copied does not exist on the target database, it is created on the target database. If the object does exist on the target database, the object is overwritten – unless the `overwrite_existing_objects` property is set to false. In that case, a create time error is generated.
- You cannot relocate or rename views.
- If the table names in the view definition are not fully qualified, those tables must exist in the database having the same name as the source view database to avoid an error.

For example, the following view definition contains a non-fully qualified view name and table name:

```
CREATE VIEW deptsals AS SELECT department_number AS department
FROM employee
HAVING salary_amount < 36000;
```

- If the table on which the view is created does not exist on the target, or is not being moved as part of the job, an error results when attempting to create the view definition on the target system.
- If you want to copy a view definition without copying the underlying table, you can use DSA. See [About Copying Views, Macros, and Stored Procedures Using DSA](#).
- Attempting to rename or relocate referenced tables results in an error when you create the view definition on the target system.

About Copying View Data

Data Mover can copy view data when underlying data is being copied from a view to a view between Teradata systems and Teradata tables. When copying data from a view, specify the `copyData="true"` attribute along with the `view` element.

The `copyData` attribute is optional, with a default value of `false`. When `copyData` is set to `true`, it may be followed with the optional `view_data_table` element.

If `view_data_table` is not specified, Data Mover copies the view data to a target view. The element `view selection = "included"` must be specified to avoid an exception at job creation time. For example:

```
<views>
  <view selection="included" copyData="true">
    <name>deptsals</name>
    <database>MyDb</database>
  </view>
</views>
```

In the example here, the data from the view `deptSals` is selected using `"SELECT *from MyDb.deptSals"` and loaded to the view `MyDb.deptSals` on the target. If the view `MyDB.deptSals` does not exist on the target, it is created using the source view definition. If the view `MyDB.deptSals` exists on the target, it is overwritten using the source view definition.

If `view_data_table` is specified, `target_table` indicates the table to which the underlying view data is copied and `target_database` indicates the database in which the target table exists. If the table does not already exist on the target, Data Mover creates the table. If the table already exists on the target, the view data is loaded to it. For example:

```
<views>
  <view selection="included" copyData="true">
    <name>deptsals</name>
    <database>MyDb</database>
    <view_data_table>
      <target_table>TargetTable</target_table>
      <target_database>TargetDb</target_database>
    </view_data_table>
  </view>
</views>
```

In the example here, the data from the view `deptSals` is selected using `"SELECT * from MyDb.deptSals"` and loaded to the table `TargetDb.TargetTable`. If `TargetDb.TargetTable` does not exist on the target, Data Mover creates the target table.

When Data Mover creates a target table, the table has the following characteristics:

- Is a multiset table

- Is created with the same column names and column types as the view
- Is a no-fallback table without journaling
- Does not contain a secondary index

In addition, if the view whose data is being copied contains a Primary Index column, that column is chosen as the Primary Index of the target table. If the view does not contain a Primary Index column, the first column in the view is chosen as the Primary Index.

When copying view data to a target view, the following apply:

- The source view must not reference more than one table. If the view has multiple reference tables, a runtime exception occurs.
- Any `compare_ddl` element is ignored because Data Mover creates the target view definition using the source view definition.
- The table referenced by the source view must exist on the target or must be moved with the job.

NOTICE

When copying view data to a target view, the source view data is first copied to a target staging table and the target view, if one exists, is overwritten with the source view definition. Then the source view data is copied from the staging table to the target view. It is possible that the target reference table could include null values if the view does not include all of the columns of that reference table. Therefore, exercise care when using this feature.

The following rules apply to the use of load utilities when copying view data to either a target table or view:

- Do not use DSA as the load utility if a source staging table is not used.
- Specifying DSA without using a source staging table or `TPTAPI_LOAD` as the `force_utility` value when copying view data results in a create time error.
- View data is copied using the `TPTAPI_UPDATE` operator unless `force_utility` is specified as `TPTAPI_STREAM`, `JDBC`, or `DSA` with source staging table.
- When view data is being copied along with other tables, the view data is copied using the `TPTAPI_UPDATE` operator but the other data tables can be copied using the `TPTAPI_LOAD` operator. This happens if `force_utility` is unspecified and source staging table is not used, or if `force_utility` is specified as `TPTAPI`.

About Copying Views, Macros, and Stored Procedures Using DSA

In Data Mover, you can copy views, macros, and stored procedures definition, and create those objects on the target system using DSA. It does not required the underlying dependent object of the copied objects in the target system. You can also use SQL instead of DSA, but it requires the underlying dependent object of the copied objects in the target system.

To copy these objects using DSA, specify the `force_utility` element value as `dsa` in the job.

For example:

```

<dmCreate>
...
  <force_utility>dsa</force_utility>
...
  <views>
    <view selection="included">
      <name>view1</name>
      <database>database1</database>
    </view>
  </views>
  <macros>
    <macro selection="included">
      <name>macro1</name>
      <database>database1</database>
    </macro>
  </macros>
  <stored_procedures>
    <stored_procedure selection="included">
      <name>storedprocedure1</name>
      <database>database1</database>
    </stored_procedure>
  </stored_procedures>
...
</dmCreate>

```

If you do not specify the `force_utility` element value, then it uses SQL for copying objects.

About Force Target Staging Table when Copying View Data

When copying a view to another table, you can force Data Mover to create a target staging table for the view data by setting `force_target_staging_table` to true.

```

<view selection="included" copyData="true">
  <name>deptsals</name>
  <database>MyDb</database>
  <view_data_table>
    <target_table>TargetTable</target_table>
    <target_database>TargetDb</target_database>
  </view_data_table>
  <force_target_staging_table>true</force_target_staging_table>
</view>

```

About Using a Target Staging Database When Moving Underlying Data from a View

If the target table to which the data is being moved exists on the target, Data Mover creates a target staging table to which data is first moved. If copying data to a target view, data is always moved to a target staging table. After data is successfully moved to a target staging table, data from the target table or view is deleted and an INSERT/SELECT or MERGE is performed to move data from the target staging to the target table or view. This is very similar to providing a staging database when moving regular data tables.

In the example here, TargetDb_Staging is used as a target staging database.

```
<views>
  <view selection="included" copyData="true">
    <name>deptsals</name>
    <database>MyDb</database>
    <view_data_table>
      <target_table>TargetTable</target_table>
      <target_database>TargetDb</target_database>
    </view_data_table>
    <staging_database>
      <name>TargetDb_Staging</name>
    </staging_database>
  </view>
</views>
```

About Using a Source Staging Database to Partially Move Underlying Data from a View Using DSA

A view can be partially moved using DSA with a source staging table, and can be moved to either a target table or a target view. The underlying data first copies to the source staging table, then DSA moves the data from the source staging table to the target system. To use a specified source staging table, you must set `use_source_staging_table` to `true` at the object level in the create XML file.

The following XML demonstrates using `source_db` to move a view to a target table:

```
<views>
  <view selection="included" copyData="true">
    <name>deptsals</name>
    <database>MyDb</database>
    <view_data_table>
      <target_table>TargetTable</target_table>
      <target_database>TargetDb</target_database>
    </view_data_table>
```

```

        <use_source_staging_table>true</use_source_staging_table>
        <source_staging_database>
            <name>source_db</name>
        </source_staging_database>
        <sql_where_clause>WHERE c1 > 500</sql_where_clause>
        <key_columns>
            <key_column>c1</key_column>
        </key_columns>
    </view>
</views>

```

The following XML demonstrates using source_db to move a view to a target view:

```

<views>
    <view selection="included" copyData="true">
        <name>deptsals</name>
        <database>MyDb</database>
        <use_source_staging_table>true</use_source_staging_table>
        <source_staging_database>
            <name>source_db</name>
        </source_staging_database>
        <sql_where_clause>WHERE c1 > 500</sql_where_clause>
        <key_columns>
            <key_column>c1</key_column>
        </key_columns>
    </view>
</views>

```

About Comparing DDL When Moving Underlying Data From a View

When moving data from a view using copyData="true" you can compare the view columns with the columns of the target table using the compare_ddl feature.

Note:

The compare_ddl feature is not used to compare the definition of the source and target views. It is used to compare the columns of the source view with the columns of the target table.

When copying time series tables, compare_ddl also compares the primary time index which includes the following fields:

- TSFlags
- timeZero
- TimeBucketUnit

- TimeBucketValue

To copy time series tables to a non-time series table, set `compare_ddl` to `false`, otherwise the job fails.

In order to compare the source view columns with the target table columns, the `compare_ddl` element needs to be specified as `true`. This is optional with a default value of `true`.

```
<views>
  <view selection="included" copyData="true">
    <name>deptsals</name>
    <database>MyDb</database>
    <view_data_table>
      <target_table>TargetTable</target_table>
      <target_database>TargetDb</target_database>
    </view_data_table>

    <staging_database>
      <name>TargetDb_Staging</name>
    </staging_database>

    <compare_ddl>true</compare_ddl>
  </view>
</views>
```

When copying view data, the `compare_ddl` element is not considered if the target data table does not exist on the target system or if copying view data to target view.

About Partial Copying of View Data

Partial data copy can be performed from a source view to a target table or view by specifying a `WHERE` clause and key columns. This is very similar to partial copy of data tables.

The following example shows a partial data copy from a source view:

```
<views>
  <view selection="included" copyData="true">
    <name>deptsals</name>
    <database>MyDb</database>
    <view_data_table>
      <target_table>TargetTable</target_table>
      <target_database>TargetDb</target_database>
    </view_data_table>

    <staging_database>
      <name>TargetDb_Staging</name>
    </staging_database>
```

```

        <compare_ddl>true</compare_ddl>

        <sql_where_clause><![CDATA[ WHERE "deptsals"."employee_number"
= 1]]></sql_where_clause>
        <key_columns>
            <key_column>employee_number</key_column>
        </key_columns>
    </view>
</views>

```

In the example here, a SELECT statement with a WHERE clause is used to select data from the view where "employee_number" = 1 and load the partial data to the target table. Key columns must be specified when a SQL WHERE clause is specified.

Note:

A view definition or data from a view can be copied without copying the tables associated with the view.

When performing a partial copy of view data to a target view, the qualifying rows are first copied to a staging table. After the rows are copied to the staging table, the qualifying records in the WHERE clause are copied from the staging table to the target view. This process is similar to a partial table copy when the target table exists with data.

About Query Band

The query band is a set of user-defined name-value pairs used to uniquely identify a session or transaction in the Teradata Database. Uses of the query band to tag a session or transaction include the following:

- Grouping requests into arbitrary jobs for accounting and control purposes
- Asserting a proxy user or proxy role for a trusted session
- Debugging performance problems
- Defining system variables

Examples of the query band definition:

If query band definition is	Because it
valid	follows syntactical rules: Job=payroll; Userid=aa1000000;Jobsession=1122;
invalid	uses the = character in the name: Job=payroll; User=id=aa1000000;Jobsession=1122;

The table here lists what happens in certain query band scenarios.

Scenario	Description
The query band contains = ; \u0000 (the NULL character)	Returns an error
The query band exceeds 2048 characters, not counting white space padding	Returns an error
The query band definition repeats a name in a name-value pair	Returns an error
A name-value pair does not end with a ;	Returns an error
The name is more than 128 characters	Returns an error
The value is more than 256 characters	Returns an error
The value is 0 characters	Allows the query band definition, but does not pass a value for the associated name

Note:

For detailed information about the query band feature using valid key-value pair, see *Teradata® Database SQL Data Definition Language - Syntax and Examples*, B035-1144.

Using Query Band

Use query band to tag sessions or transactions in the Teradata Database. In Data Mover, you can enable or disable the default query band feature using the daemon configuration parameter **job.default.queryband.enabled**. If the default query band feature is enabled, Data Mover uses the default daemon-level query band **job.default.queryband** property value such as the following example:

```
ApplicationName=DM;Version=16.20;
```

You can update the default query band property value with the enabled or disabled command **job.default.queryband.enabled**. This property value is used in the command parameters **list_configuration** and **save_configuration**.

You can provide a query band value at the job level using the **query_band** xml tag or the REST **queryBand** value when running create, move, or edit job. If the default query band feature is disabled, you must provide the query band value at job level to use query band; otherwise, query band not used.

1. Use the appropriate query band level steps to set the query band property values:

Query Band Level	Steps
Job Level	<p>At the job level, query band has a higher priority than the default query band at daemon level.</p> <ol style="list-style-type: none"> Create a job definition with a list of objects to move, appending the <code><query_band></query_band></code> element before the final closing tag <code></dmCreate></code>. Enter the query band information as a set of semicolon-separated name value pairs between the query band elements as in the following example: <code><query_band>Job=payroll;Userid=aa1000000;Jobsession=1122;</query_band></code> Add or modify the <code><query_band></code> value using the create, move, or edit job commands.
Daemon Level	<p>At the daemon level, query band settings are applicable to all jobs. Use the <code>list_configuration</code> or <code>save_configuration</code> commands to update property values.</p> <ol style="list-style-type: none"> To enable the default query band feature, set the <code>job.default.queryband.enabled</code> property as true. Set the default query band property, <code>job.default.queryband</code>, with the appropriate values.

About Database Client Encryption

You can encrypt data during a transfer. Data can be encrypted at the system, job, and object level by using either the system level property `job.databaseClientEncryption`, or the job and object level property `db_client_encryption`.

Client encryption behaves differently depending on the `force_utility` selected:

Utility	Behavior
TPTAPI	Data is encrypted during transfer.
QueryGrid or JDBC	Only the connection between the source and target system is encrypted, not the data.

Scope Levels and Priority Logic

The following table shows the objects included at each scope level, and the priority logic applied if you specify different values at the scope levels. If you set the database client encryption flag at more than one scope level, the smallest scope has the highest priority. For example, if you set `job.databaseClientEncryption` to true at the system level, but false at the job level, the data transfer for this is not encrypted.

Scope Level	Objects Compared	Priority
Object	The specified table or view	Highest
Job	All tables and views in the job	High

Scope Level	Objects Compared	Priority
Daemon	All tables and views processed by a specific daemon	Low

About Supporting TLS 1.2 with SQL-Engine

Data Mover supports TLS 1.2 to encrypt communication between itself and SQL-Engine. The following utilities support TLS 1.2.

Utility	Behavior
TPTAPI	Data movement between TPTAPI (running on Data Mover servers) and Source/Target SQL-Engine is encrypted.
JDBC	All communication and data movement by JDBC (running on Data Mover servers) and Source /Target SQL-Engine is encrypted. This includes calling DBC Views, creating or dropping tables on target, copy stats, and others.

To enable TLS 1.2 on Data Mover server, the two properties `tpt.connection` and `jdbc.connection` are added in the daemon level (daemon configuration parameter). Users can specify extra parameters that are passed to TPTAPI and JDBC which establishes TLS connection to SQL-Engines. Data Mover passes the parameters exactly as specified in the configuration to the respective utilities. It does not check, parse, or modify the user input.

Note:

Parameters are separated with a semi-colon (;) in `tpt.connection`, and with a comma (,) in `jdbc.connection`.

A subset of parameters to allow TLS 1.2 are `sslmode`, `tdmstport`, `tdmstlsport`, `sslca`, `sslcapath`, and `sslprotocol`. Please refer to the documentation on TPTAPI and JDBC to configure the exact parameters.

In daemon configuration, for `tpt.connection` and `jdbc.connection`, Data Mover allows user to specify a system specific value and a 'default' value for all systems. The system specific value has precedence of default value. For example,

```
<property>
  <key>tpt.connection</key>
  <value>sslmode=allow;</value>
  <value system="systemA">sslmode=required; tdmstport=1025;
tdmstlsport=443; sslcapath=/etc/ssl/mycerts</value>
</property>
```

Here, `sslmode=required` and three other parameters are used for TPTAPI connection when `systemA` is a source or target; and `sslmode=allow` is used for all the other systems.

User can specify both properties through the command line (`datamovelist/save_configuration`) as well as through REST API.

Note:

Support for DSA with TLS 1.2 encryption of user data is now available. Refer the *Teradata® Data Mover Installation, Configuration, and Upgrade Guide for Customers*, B035-4102 for more information about how to configure the encryption of all user data from source to target. This configuration applies to all DM or DSA jobs that are executed.

DDL Comparison

The DDL comparison feature compares the DDL of the source table or view columns with the DDL of the target table or columns prior to copying the table or view. If the table does not already exist on the target system, the comparison is skipped.

When the feature is enabled, the comparison takes place every time the create or move command is used to create a job, or the start command is used to start a job.

DDL Comparison for Tables

The DDL comparison feature for tables compares the number of columns, MAPS properties, and Primary Time Index (PTI) properties between the source and target table:

Note:

If any of the following values are not the same for the source and target tables, the job fails.

- Number of columns with the following values for each column:
 - Auto column
 - Character type
 - Compressible
 - Constraint
 - Constraint count
 - Decimal fraction digits
 - Decimal total digits
 - Default value
 - Format
 - Identity column type
 - Length
 - Name
 - Nullable
 - Stored procedure parameter type

- Schema
- Storage format
- TS column type
- Type
- UDT name
- Upper case flag
- MAPS properties
 - The map and colocate properties do not have to match when they exist on both the source and target tables.
 - If map and colocate parameters are provided in a job, those parameters must match the target table map and colocate parameters.
- PTI properties
 - Both the source and target tables must be PTI tables.
 - PTI properties (timezero, tsFlags, and timeBucketValue) must match on both the source and target table.

DDL Comparison for Views

DDL comparison for a view compares the columns of the source view with the columns of the source table. It does not compare the definitions of the source and target views. To compare DDL when copying data from a view, the copyData attribute for the view must be set as true.

Scope Levels and Priority Logic

The following table shows the objects included at each scope level, and the priority logic applied if you specify different values at the scope levels. If you set the DDL comparison flag at more than one scope level, the smallest scope has the highest priority. For example, if you set the <compare_ddl> element to true at the database level but false at the job level, DDL comparison for all of the tables and views in the specified database is performed.

Scope Level	Objects Compared	Priority
Object	The specified table or view	Highest
Database	All tables and views in the specified database	Higher
Job	All tables and views in the job	High
Daemon	All tables and views processed by a specific daemon	Low

If none of the values for the scope levels are specified, the default logic applies:

Objects Copied	Default Logic
Individual tables and views	True. DDL comparison is performed.
An entire database	False. DDL comparison is not performed.

Setting DDL Comparison at the Object, Database, or Job Level

Use the `compare_ddl` element in the job creation XML to specify that DDL comparison be performed for the individual object, for all tables and views in a database, or for all tables and views in the job.

The `compare_ddl` flag is always optional at the object, database or job level; you do not need to explicitly set a value. By default, DDL comparison is enabled when you specify an individual table or view to be copied, and disabled for tables and views when copying an entire database.

1. Open the `parameters.xml` file that has the job definition information for the job.
2. Add a `compare_ddl` element in the appropriate location in the job creation XML, and specify the desired value:

Option	Description
true	Perform DDL comparison for the individual object or for all tables and views in the database or in the job.
false	Do not perform DDL comparison for the individual object or for all tables and views in the database or in the job
unspecified	Use the default value unless a different higher priority value is specified.

The following examples illustrate the placement of the `compare_ddl` element, according to the scope level at which you want to set the DDL comparison flag.

Table example: DDL comparison is performed for Table1 (by default) and not performed for Table2.

```
<table selection="included">
  <name>Table1</name>
</table>
<table selection="included">
  <name>Table2</name>
  <compare_ddl>false</compare_ddl>
</table>
```

View example: DDL comparison is performed when copying the view. Note that the `copyData` element must also be set to `true`.

```
<views>
  <view selection="included" copyData="true">
    <name>deptsals</name>
    <database>MyDb</database>
    <view_data_table>
      <target_table>TargetTable</target_table>
      <target_database>TargetDb</target_database>
    </view_data_table>
  </view>
</views>
```

```

    <compare_ddl>true</compare_ddl>
  </view>
</views>

```

Database example: DDL comparison is performed for all tables and views in the database.

```

<database selection="all">
  <name>MyDB</name>
  <compare_ddl>true</compare_ddl>
</database>

```

Tables in a database example: DDL comparison is performed on tables in the MyDB database other than the MyDB.PPIOrders table when an entire database is copied.

```

<database selection="all">
  <name>MyDB</name>
  <compare_ddl>true</compare_ddl>
  <table selection="included">
    <name>PPIOrders</name>
    <compare_ddl>false</compare_ddl>
  </table>
</database>

```

Job example: DDL comparison is performed for all tables and views in the job.

```

<job_name>MyJob</job_name>
...
<overwrite_existing_objects>true</overwrite_existing_objects>
<force_utility>tptapi</force_utility>
<compare_ddl>true</compare_ddl>
<database>
...
</database>

```

Setting DDL Comparison at the Daemon Level

Use the `compare_ddl` element in the job creation XML to specify that DDL comparison be performed for the individual object, for all tables and views in a database, or for all tables and views in the job.

The `compare_ddl` flag is always optional at the object, database or job level; you do not need to explicitly set a value. By default, DDL comparison is enabled when you specify an individual table or view to be copied, and disabled for tables and views when copying an entire database.

1. Run the `list_configuration` command.
The command writes its output to a configuration XML file (by default: `./configuration.xml`)
2. Open the configuration XML file.
3. For the key `daemon.default.compareDDL.enabled` under the property element, specify the desired value:

Option	Description
true	Perform DDL comparison for all of the tables and views copied using this daemon.
false	Do not perform DDL comparison for all of the tables and views copied using this daemon.
unspecified	Default

Daemon level example: Enabling DDL comparison for all tables and views copied using that daemon.

```
<property>
  <key>daemon.default.compareDDL.enabled</key>
  <value>true</value>
  <description>Purpose: Enable/Disable the default compareDDL behavior at the
daemon level. Default value unspecified.</description>
</property>
```

Ignore Missing Source Objects

Job fails to run if any source object which is used to create this job does not exist at run time. Ignore missing source objects feature allows job execution to not fail because of missing source objects. This feature has the following restrictions:

- All source objects must still exist when creating a new job or editing an existing job. This restriction applies to the create, move, and edit commands.
- All source objects must still exist when using the parameterized start command with save_changes option.
- Feature does not apply to objects within a database that is being copied in a full database copy. Missing source objects in that scenario are ignored regardless of this setting.
- Feature does not apply to jobs using the frozen job steps option. Jobs using frozen job steps option fails even if source object is missing. Use the edit command to remove the missing source objects from the job in this scenario.
- All source objects must exist when running the update job steps command to update a job using frozen job steps.

To enable this feature, change the following property value to "TRUE" in configuration.xml

```
<property> <key>ignore.missing.source.objects</key> <value>TRUE</
value> <description>
```

Purpose: Defines whether to fail a Non Frozen job if any source object is missing. This property only applies to Non Frozen jobs and missed objects are not part of entire databases move. Default value is false to preserve existing behavior.

```
</description> </property>
```

About Copying Entire Databases

You can copy all the objects from a source to a target database without specifying each object in the database when using Teradata DSA. This is a useful alternative to specifying a copy of individual tables, especially when the database has a large number of objects. When copying the entire database to a target, you can do any or all of the following:

- Exclude specific tables from being copied
- Rename tables to be copied in the target
- Copy partial table data
- Copy join and hash indexes
- Copy triggers
- Enable the compare DDL feature for all tables to be copied
- Disable the compare DDL feature for selected tables to be copied
- Rename and relocate hash and join indexes
- Copy entire databases and all their children
- Exclude child databases from being copied
- Rename tables in a child database
- Copy specific tables in a child database
- Copy to a different target database
- Copy a parent database with its children, but relocate a child database to a different database on the target

Rules and Restrictions

- If a job copies the entire database using Teradata DSA and the database has tables with foreign keys, the target tables are created without foreign keys.
- You can only copy an entire database between Teradata Database systems using the DSA utility.
- Teradata PT API operators and Teradata JDBC utility do not support copying an entire database.
- The specified target database must exist and have adequate space.
- When using DSA to copy an entire database on which statistics have been collected, the statistics of all objects within the database are copied regardless of the copyStats attribute.

Copying an Entire Database

The following applies when copying an entire database:

- You can copy all tables and views from a source to a target database using Teradata DSA.
- If copying a table that already exists on the target, the target table is overwritten. If the target database contains tables that do not exist on the source or the tables are not being copied, those tables are maintained on the target. The exception is when using DSA to copy data to a target database with a version earlier than 16.20, in that scenario the tables are deleted.

- DSA automatically copies statistics collected on all tables in the source database to the target database.

Note:

Full databases from newer to older versions of Teradata Database cannot be copied. This is a restriction with DSA and other utilities that an entire database cannot be moved. Teradata recommends that the user must upgrade the target database to a newer or same version as the source database. If upgrade is not possible the alternative solution is to utilize other available utilities to move objects individually.

1. Open the `parameters.xml` file that has the job definition information for the job.
2. Add the `selection="included"` attribute to the database element.

In the following example, all objects in the source database `DBName` are copied to the target:

```
<database selection="included">
  <name>DBName</name>
</database>
```

Excluding Tables

You can exclude one or more tables from being copied from the source to the target when you copy an entire database using DSA.

If the `EXCLUDE TABLE` clause is too long for the SQL buffer, an error occurs. The buffer length limit depends on how much data already exists in the SQL buffer. If an error occurs, modify the `EXCLUDE TABLE` list by reducing its size before resubmitting the job.

1. Open the `parameters.xml` file that has the job definition information for the job.
2. Under the database name, add the `selection="excluded"` attribute to the table element.
3. Add a `<name>` element and enter the name of the table to be excluded.

In the following example, all tables except `DBName.Orders` and `DBName.Accounts` are copied to the target.

```
<database selection="included">
  <name>DBName</name>
  <table selection="excluded">
    <name>Orders</name>
  </table>
  <table selection="excluded">
    <name>Accounts</name>
  </table>
</database>
```

Renaming Tables

You can rename one or more tables on the target when you copy an entire database.

1. Open the `parameters.xml` file that has the job definition information for the job.
2. Under the database name, add the `selection="included"` attribute to the `table` element.
3. Add a `name` element and enter the current name of the table.
4. Add a `target_name` element and specify the new name for the table.

In the following example, the `DBName.Orders` table is renamed to `DBName.Orders_Newname` and `DBName.Accounts` is renamed to `DBName.Accounts_2` on the target.

```
<database selection="included">
  <name>DBName</name>
    <table selection="included">
      <name>Orders</name>
      <target_name>Orders_Newname</target_name>
    </table>
    <table selection="included">
      <name>Accounts</name>
      <target_name>Accounts_2</target_name>
    </table>
</database>
```

You can specify a staging database for the table to be renamed. In the following example, the `DBName.Orders` table is copied to the `stage1` staging database and renamed to `stage1.Orders_Newname`.

```
<database selection= "included">
  <name>DBName</name>
    <table selection="included">
      <name>Orders</name>
      <staging_database>
        <name>stage1</name>
      </staging_database>
      <target_name>Orders_Newname</target_name>
      <staging_database>
    </table>
</database>
```

Join and Hash Index Version Considerations

You can copy join and hash indexes as part of a full database copy using DSA, but not as part of a full database relocation where the target database name is not the same as the source database name.

Copying a database with join or hash indexes on the target, but no corresponding indexes on the source, re-creates the target indexes after the entire database is copied, if any indexes exist on the target database.

To copy join and hash indexes when copying an entire database using DSA, set the optional **database.copy.joinIndex** configuration property to true as shown in the following sample `configuration.xml`:

```
<property>
  <key>database.copy.joinIndex</key>
  <value>true</value>
  <description>Purpose: If set to true, all Join Index and Hash Index
associated with a table will automatically be copied to the target when
copying the table between DBS version 13.0 or higher. This parameter is
only used to support previous functionality of Data Mover. The default is
false.</description>
</property>
```

Note:

DSA automatically copies join or hash indexes when the source or target database version is earlier than 16.20 regardless of the `database.copy.joinIndex` value.

Copying Join and Hash Indexes

When copying entire databases using DSA, join and hash indexes on the source database are not automatically copied to the target when either relocating the database or when **database.copy.joinIndex** is not set to true. You must copy join and hash indexes explicitly during a database relocation or during a database copy when **database.copy.joinIndex** is set to false using the following instructions:

1. Open the `parameters.xml` file that has the job definition information for the job.
2. Create an `indices` section.
3. Add the `selection="included"` attribute to an index element.
4. Specify the name of the join or hash index using a `name` element.
5. Specify the index type using an `index_type` element. The possible values are `HASH_INDEX` and `JOIN_INDEX`.

Note:

DSA automatically copies join or hash indexes when the source or target database version is prior 16.20. In this instance, creating an additional `indices` section is not necessary. In addition, when using DSA on databases with versions earlier than 16.20, extra indices are not supported.

In the following example, the hash index `DBName.Orders_hi` associated with the `DBName.Orders` table is copied to the target when copying the database `DBName`.

```

<database selection="included">
  <name>DBName</name>
</database>
  <indices>
    <index selection="included">
      <name>Orders_hi</name>
      <index_database>DBName</index_database>
      <index_type>HASH_INDEX</index_type>
    </index>
  </indices>

```

Copying Triggers

Although DSA does not automatically copy triggers from the source when copying entire databases, you can specify triggers for an associated table in the target.

The following results in an error when copying triggers:

- The source table associated with the trigger is excluded from the tables to be copied
 - The table associated with the trigger is renamed or relocated on the target
 - You do not use AFTER as the value of the `action_time` tag.
1. Open the `parameters.xml` file that has the job definition information for the job.
 2. Create a `triggers` section.
 3. Add the `selection="included"` attribute to a `trigger` element in the section.
 4. Specify the names for the `database` and `subject_table_database` elements.
 5. Specify the table name in the `table` element.
 6. Specify the trigger name in the `name` element.
 7. Use the `action_time enabled="YES">AFTER</action_time>` tag to indicate when the trigger occurs.

In the following example, the trigger `DBName.RaiseTrig` is copied to the target.

```

<database selection="included">
  <name>DBName</name>
</database>
  <triggers>
    <trigger selection="included">
      <database>DBName</database>
      <subject_table_database>DBName</
subject_table_database>
      <table>PPIOrders</table>
      <name>RaiseTrig</name>
      <action_time enabled="YES">AFTER</action_time>
    </trigger>
  </triggers>

```

Enabling Compare DDL for All Tables

Although Data Mover does not automatically compare DDL of tables when copying the entire database, you can enable compare DDL for all database tables being completely or partially copied or renamed.

1. Add the `compare_ddl` element.
2. Specify `true` as the element value.

In the following example DDL comparison is performed on all tables in the MyDB database.

```
<database selection="all">
  <name>MyDB</name>
  <compare_ddl>true</compare_ddl>
</database>
```

Disabling Compare DDL for Selected Tables

Data Mover does not automatically compare DDL tables when copying the entire database. You can enable DDL comparison for database tables being completely or partially copied or renamed, while excluding DDL comparison for certain tables.

1. Add the `compare_ddl` element.
2. Specify `true` as the element value at the database level.
3. Add the `selection="included"` attribute to a table element and specify the table name.
4. For the table to be excluded, add the `compare_ddl` element and specify `false` as the element value.

In the following example, compare DDL is performed on tables in the MyDB database. However, compare DDL is not performed for the MyDB.PPIOOrders table.

```
<database selection="all">
  <name>MyDB</name>
  <compare_ddl>true</compare_ddl>
  <table selection="included">
    <name>PPIOOrders</name>
    <compare_ddl>false</compare_ddl>
  </table>
</database>
```

In the following example, compare DDL is performed on tables in the MyDB database and the MyDB.PPIOOrders table is renamed to MyDB.PPIOOrders_NewName on the target. Compare DDL is not performed for the MyDB.PPIOOrders_NewName table.

```
<database selection="all">
  <name>MyDB</name>
  <compare_ddl>true</compare_ddl>
  <table selection="included">
    <name>PPIOOrders</name>
    <target_name>PPIOOrders_NewName</target_name>
    <compare_ddl>false</compare_ddl>
  </table>
</database>
```

Copying Entire Databases and All Their Children

1. Add the selection="all" attribute to the database element.

In the following example, the MyDatabase database and all its children are copied to the target.

```
<database selection="all">
  <name>MyDatabase</name>
</database>
```

Excluding Child Databases

1. Add the selection="excluded" attribute to the database element.
2. Specify the name of the child database.

In the following example, the MyDatabase database and all its children except for MyChildDB are copied to the target.

```
<database selection="all">
  <name>MyDatabase</name>
    <database selection="excluded">
      <name>MyChildDB</name>
    </database>
</database>
```

Renaming Tables in a Child Database

1. Add the selection="included" attribute to the table element.
2. Specify the name of the source table.
3. Specify the renamed target table with the target_name element.

In the following example, MyDatabase and all its children are copied to the target, but the MyDatabase_child.ORDERS table is renamed to MyDatabase_child.ORDERS_NEWNAME on the target.

```
<database selection="all">
  <name>MyDatabase</name>
    <database selection="included">
      <name>MyDatabase_child</name>
        <table selection="included">
          <name>ORDERS</name>
          <target_name>ORDERS_NEWNAME</target_name>
        </table>
      </database>
    </database>
```

Copying Specific Tables

You can copy an entire database and its children, but specify that only specific tables in one of the child database be copied.

1. Add the selection="unselected" attribute to the database element.

2. Specify the name of the child database.
3. Add the selection="included" attribute to the table element.
4. Specify the name of the table of the child database to be copied.

In the following example, the MyDatabase database and all its children except MyDB_Child are copied to the target. However, the MyDB_Child.call_hist table is copied to the target.

```
<database selection="all">
  <name>MyDatabase</name>
    <database selection="unselected">
      <name>MyDB_Child</name>
        <table selection="included">
          <name>call_hist</name>
        </table>
      </database>
    </database>
```

Relocating an Entire Database

You can relocate all objects from a source database to a different target database. To copy all objects in a source to a different target database, add the target_database element and specify the name of the target database. You can only specify target_database if you specify selection = "included". Specifying the target_database element when selection= "all" results in a create time error.

The target database must exist before creating the job. Data Mover does not create the database on the target system.

1. Add the target_database element.
2. Specify the name of the target database.

In the following example, objects in the database MySourceDB are copied to the target database MyTargetDB.

```
<database selection="included">
  <name>MySourceDB</name>
  <target_database>
    <name>MyTargetDB</name>
  </target_database>
</database>
```

When relocating an entire database, Teradata DSA does not automatically copy the join and hash indexes from the source database. The join or hash index to be copied to the target needs to be specified in the XML. The index is created on the relocated table on the target. The join or hash index itself is not relocated.

Note:

When using DSA on databases with versions earlier than 16.20, specifying an additional indices section in the job is not supported. A separate job must be created to copy only indices when using DSA to relocate an entire database.

In the following example, hash index MyDB.orders_hi is copied to the target database MyDB. Because the associated table MyDB.Orders is being relocated to MyTargetDB, the hash index is created on the table MyTargetDB.Orders on the target.

```
<database selection="included">
  <name>MyDB</name>
  <target_database>
    <name>MyTargetDB</name>
  </target_database>
</database>
  <indices>
    <index selection="included">
      <name>Orders_hi</name>
      <index_database>MyDB</index_database>
      <index_type>HASH_INDEX</index_type>
    </index>
  </indices>
```

Relocating a Child Database

You can relocate a child database to a different target database when copying the entire parent database and its children.

1. Add the selection="all" attribute to the database element for the parent.
2. Specify the parent database name.
3. Add the selection="included" attribute to the database element for the child database.
4. Specify the child database name, as well as the target_database element and the target database name.

In the following example, the parent database MySourceDB is copied to MySourceDB, but the child database MyChildDB is copied to the database MyTargetDB.

```
<database selection="all">
  <name>MySourceDB</name>
  <database selection="included">
    <name>MyChildDB</name>
    <target_database>
      <name>MyTargetDB</name>
    </target_database>
  </database>
</database>
```


About Replacing Databases

When you copy all objects in a Teradata Database by specifying `<database selection="included">`, Data Mover copies all of the tables from the source to the target, but leaves any additional tables that exist in the target database intact. For example, if the source database contains tables A, B, and C and the target contains tables A, B, C, and D, tables A, B, and C are copied from the source to the target database, and table D remains on the target database.

The database replacement feature enables you to completely overwrite or "replace" a database on the target system by eliminating all of the objects in the target database and replacing them with those on the source. Any additional tables that existed on the target database but not on the source are removed. You replace a database by setting an optional `replaceDatabase` attribute as `true`, as shown here:

```
<database selection="included" replaceDatabase="true">
  <name>MyDatabase</name>
</database>
```

The default value for the `replaceDatabase` attribute is `"false"` if not specified. If the attribute is set as `<replaceDatabase="false">` or is not specified at all, the tables on the target that do not exist on the source remain on the target after the entire database is copied.

Note:

When using DSA and the source or target database is earlier than Teradata Database 16.20, Data Mover overwrites the target database regardless of the `replaceDatabase` value.

Excluding Tables When Replacing a Database

You can exclude a table when replacing a database by specifying the table name. Excluding a table that exists on the target does not remove the existing target table.

If the `EXCLUDE TABLE` clause is too long for the SQL buffer, an error occurs. The buffer length limit depends on how much data already exists in the SQL buffer. If an error occurs, modify the `EXCLUDE TABLE` list by reducing its size before resubmitting the job. Another workaround is to select all the tables instead of selecting a full database copy, so the exclude clause is not included.

1. Set `<replaceDatabase="true">` to replace the specified database.
2. Set `<table selection="excluded">` and then specify the table name.

In the following example the target database is replaced with the source database but the `Customers` table is not copied to the target.

Note:

If the `Customers` table already exists on the target, excluding it in the XML as shown here does not remove it from the target.

```
<database selection="included" replaceDatabase="true">
  <name>MyDatabase</name>
  <table selection="excluded">
    <name>Customers</name>
  </table>
</database>
```

Replacing and Relocating a Database

When you replace a database, you may also choose to relocate it.

1. Set `replaceDatabase` to `true`, and specify the name of the source database.
2. Under the `target_database` element, specify the name of the location where you want to copy the source database.

In the following example, all objects in the `MyTargetDB` database are eliminated and objects from the source `MySourceDB` database are copied to the target database `MyTargetDB`. Thus `MyTargetDB` is replaced with `MySourceDB`.

```
<database selection="included" replaceDatabase="true">
  <name>MySourceDB</name>
  <target_database>
    <name>MyTargetDB</name>
  </target_database>
</database>
```

Replacing a Database and its Children

1. Set `database selection="all"` to copy the entire database.
2. Set `replaceDatabase` to `true`.

In the following example, the value for `replaceDatabase` applies to `MyDatabase` and all of its children. All objects in `MyDatabase` and its children are copied to the target.

```
<database selection="all" replaceDatabase="true">
  <name>MyDatabase</name>
</database>
```

Overriding Replacement of a Child Database

You may want to override the `replaceDatabase` attribute for a particular child database when copying a database and all of its children.

1. Set the `replaceDatabase` to `true` for the parent database and all of its children.
2. Set the `replaceDatabase` to `false` for the child database that you do not want to replace, and provide its name.

Note:

The `replaceDatabase="false"` attribute is ignored when the entire database is not being copied.

In the example here, `replaceDatabase` is `true` for the database `MyDatabase` and all of its children other than `MyDb_2`.

```
<database selection="all" replaceDatabase="true">
  <name>MyDatabase</name>
  <database selection="included" replaceDatabase="false">
    <name>MyDb_2</name>
  </database>
</database>
```

About Partial Failures When Copying Tables and Databases

A number of scenarios exist in which tables and databases in a job continue to be copied after a table or database copy fails.

Utility Used	Objects Copied	Scenario	Result
TPTAPI	Tables	Table has an incorrect column name in its WHERE clause	Table copy fails. Job continues and the remaining tables are copied to the target database.
TPTAPI	Tables	Source user does not have SELECT access to one of the tables	Table copy fails. Job continues and the remaining tables are copied to the target database.

Data Mover Portlets

Data Mover Setup

The **Data Mover Setup** portlet enables the Teradata Viewpoint Administrator to add and configure management and performance settings for one or more daemons. Users can select and monitor the work of enabled daemons in the **Data Mover** portlet.

The Teradata Viewpoint Administrator can:


- Add, enable, and delete daemons
- Configure defaults for event logging, database query method, encryption, and certain job level settings
- Designate when to purge old job execution data and job definition
- Designate the users and roles that can access daemons and jobs
- Set up tables for logging Teradata Ecosystem Manager events
- Designate and configure default target databases
- Designate and configure default staging databases for tables and temporary artifacts
- Configure job timeout
- Configure retry intervals for blocked jobs and deadlocked transactions
- Configure load slots, target system restrictions, and force direction pairs

Adding or Enabling a Daemon

You must add one or more daemons in the **Data Mover Setup** portlet before you can monitor their work in the **Data Mover** portlet. You can only perform this procedure on a daemon that is currently operating. A daemon that is not currently operating cannot be saved in the **Data Mover Setup** portlet.

Note:

You can only monitor the daemon on one Viewpoint server. Monitoring the same daemon on more than one Viewpoint server is not supported and could create authentication and job permission issues. This restriction applies regardless of whether security management is currently enabled on the daemon or not.

1. Open the **Data Mover Setup** portlet.
2. Click  next to **Daemons**.
3. Enter a **Daemon Nickname**.
4. [Optional] Activate this daemon for monitoring by selecting **Enable daemon**.
5. Enter the **Daemon IP/DNS** value.
Teradata recommends using the DNS or alias rather than the IP address.

6. Enter the REST **Port** value.
7. [Optional] Select **https enabled** to enable or disable using https: for REST calls.
8. Enter the **Username** and **Password** for the daemon.
9. [Optional] Enable **QueryGrid Manager REST API** by entering the QueryGrid **Username** and **Password**.
This step is required if you want to create and monitor jobs using QueryGrid 2.x.
10. [Optional] Enable clustering when connecting to the daemon, by selecting **Enabled in portlets** and entering the **IP/DNS:Port** value as in the following example:
DM2:1443
The clustering field values must be set individually for each daemon.
11. [Optional] Select a method for sending events from the **Teradata Ecosystem Manager Logging** list.

Option	Description
NONE	Does not send events to Teradata Ecosystem Manager or to a Data Mover event table.
ONLY_REAL_TMSM	Sends events to Teradata Ecosystem Manager.
ONLY_INTERNAL_TMSM	Sends events to an internal event table that Data Mover manages.
BOTH	Sends events to Teradata Ecosystem Manager and to an internal event table that Data Mover manages.

12. [Optional] Enable Teradata Ecosystem Manager logging by entering the **Logging Frequency** value and selecting the units.
This is the rate at which Data Mover sends log events to Teradata Ecosystem Manager in units per second when copying a table.
13. [Optional] Select the data dictionary view type from the **Database Query Method** list.

Option	Description
DBC Base Views	Offers better performance but is less secure.
DBC X/VX Views	Offers better security, but has slower performance.

14. [Optional] Select the **Database to client encryption** check box to encrypt data sent from the database.
15. [Optional] Select from the **Compare DDL** list whether to compare the DDL of source tables and views with those of their existing target tables and views.
Views are compared only if their copyData attributes have been specified as true. The option you choose from this list is reflected as the default for the **Compare DDL** list in **Job Level Settings** when creating a job.

Option	Description
Unspecified	Default.

Option	Description
True	Compare the DDL of source tables and views to those of their corresponding target tables and views.
False	Do not compare the DDL of source tables and views to those of their corresponding target tables and views.

16. [Optional] Select the **Online archive** check box to set the default value for online archiving to be used.
17. [Optional] Select the **Overwrite existing objects** check box to set the default value to overwrite jobs.
18. [Optional] Select the **Disable fallback** check box to disable fallback when copying objects.
19. [Optional] Select the **Skip Fast/Multi Loaded tables** check box to skip objects that have been set to fast or multi load.
20. Click **Apply**.

Disabling a Daemon

Disabling a daemon prevents it from being available for monitoring and use in the **Data Mover** portlet, but maintains its configuration settings for future use.

1. In the **Daemons** column, click the name of the daemon you want to update.
2. In the **Daemon Setup** column, click **General**.
3. Clear the **Enable daemon** check box.
4. Click **Apply**.
☐ appears next to the name of the disabled daemon in the **Daemons** column.

Deleting a Daemon

Deleting a daemon removes it from Teradata Viewpoint so that you can no longer monitor its activity in the **Data Mover** portlet.

If you want to preserve configuration information for a daemon so that you can monitor its activity in the future, you can disable the daemon by clearing the **Enable daemon** check box in the **General** settings screen.

1. In the **Daemons** column, click the name of the daemon you want to update.
2. In the **Daemon Setup** column, click **General**.
3. Click **Delete Daemon**.
A confirmation message appears.
4. Click **Delete**.

Job Cleanup

Data Mover has an internal repository that stores job data from previously run jobs. The purge service is enabled by default and jobs are purged every 60 days. You can adjust this setting as needed based on your

business requirement. You can designate cleanup based on how old the data is, what percentage of the repository is full, or by both age and percent full. A daily cleanup check is performed at the time you specify; and if the age or space threshold is reached, the excess data is deleted from the repository. By default, Data Mover does not purge data that is newer than 5 days.

Purge by Age

When purging based on how old the data is, Data Mover deletes any job history with a run time older than the specified date period. For example, if the **Older than** setting is set to 30 days, Data Mover deletes the job history for all jobs run more than 30 days ago. If purging job definitions, all job definitions that have not been run in more than 30 days are also deleted.

Purge by Percent

When purging based on the percentage of how full the repository is, the job history is deleted in batches by time frame. Data Mover starts with deleting data older than 180 days. If more space is needed to meet the designated percentage, Data mover then purges data older than 173 days and continues to delete data in 7 day increments until the specified percentage of permanent space has been made available in the repository or until the 5 day limit is met.

Purge by Age and Percent

When purging based on both age and percentage, Data Move first deletes all data based on the specified age range. For example, if the **Older than** setting is set to 30 days, Data Mover deletes all data older than 30 days first. Data Mover then checks to see if the designated percentage of permanent space is free. If more space is needed, Data Mover continues to delete data from oldest to newest until either the specified percentage of permanent space has been reached or the 5 day limit is met.

Note:

If the purge by age setting is greater than 5 days and the percentage of free space is not met, the purge by percent feature may delete data newer than the purge by age setting.

Cleaning Up Job History

To designate job history cleanup based on age or percentage, do the following:

1. In the **Daemons** column, click the name of the daemon to configure.
2. In the **Daemon Setup** column, click **Job Cleanup**.
3. Select the **Enable job cleanup** check box to enable the settings and schedule for job cleanup.
4. Under **Delete the Following**, select one of the following:

Option	Description
Job Execution Data	Delete the job history
Job Execution Data and Job Definitions	Delete the job history and all definitions

5. Under **Delete When Data is**, select one or both of the following:

Option	Description
Older than	Delete job history based on age (days, weeks, months, or years)
Greater than	Delete job history based on percentage of total space in use

6. In the **Perform Cleanup Daily at** field, specify the time to perform the daily cleanup, or accept the default.
7. Click **Apply**.
When either the age or space usage threshold that you designate is reached, the excess data is deleted.

Permissions

A Viewpoint Administrator can assign permissions to users and roles to enable security management in the Permissions view of the **Data Mover Setup** portlet. When security management is enabled, the Viewpoint Administrator can designate the specific access types to individuals and roles, both at the daemon and at the job level.

If security management is not enabled, the selected users and roles have full access (read, write, and execute) to the daemon. A Viewpoint user who has been granted access privileges can perform any operation on any job in the **Data Mover** portlet. When **Give full daemon access to all selected users and roles** is set, it is not possible to designate certain roles or users with certain privileges but not others (such as read but not execute) or to designate job permissions without using security management.

A user must have Read access to monitor the daemon in the main **Data Mover** portlet, regardless of whether security management is enabled. For more information on granting permissions, see [Adding Permissions](#).

Access Types

Data Mover supports three different types of access privileges:

Access Type	Privileges
Read (R)	Preview jobs View status of running or started jobs
Write (W)	Create jobs Edit jobs Delete jobs Update job steps Update job priorities Edit job scripts Change permissions
Execute (X)	Run jobs

Access Type	Privileges
	Stop jobs Restart jobs Clean up jobs

A user who has write or execute access always has read access also. Therefore, the following access combinations can be designated for users and roles for a daemon or a job:

- No access
- Read access
- Read and write access
- Read and execute access
- Read, write, and execute access

For a user who does not have full access privileges, the options appear as unavailable, in grey, in the **Data Mover** portlet. For example, if a user does not have execute access, the Run command in the menu for the job appears in grey and is unavailable. For a user who does not have write access, the **New Job** button appears in grey and is unavailable.

Daemon-Level and Job Permissions

There are three options for security when security management is enabled. These options only affect the selected daemon.

Option	Description
Give full daemon access to all selected users and roles	Security management is considered disabled if this setting is selected. You cannot grant or revoke individual read/write/execute privileges to users or roles with this option. However, the user or role does still need read permissions on the daemon. For more information on granting permissions, see Adding Permissions .
Allow daemon write or execute access to be configured per user or role	<p>This daemon-level option is less restrictive than job-level permissions by allowing users access to all jobs on that daemon. For example, if this option is selected, a user with read access can perform read operations on all of the jobs on the daemon from the Data Mover portlet. Any access designated for individual jobs is disregarded when this security option is used.</p> <p>You can set read, write, and execute access for both specific users and for roles at the daemon level. A user who has write or execute access automatically has read access as well.</p>
Allow user and role access to be further restricted per job	<p>This job-level setting is more restrictive. The access specified for the daemon as well as the access specified for individual jobs are considered. For example, a user with read access on the daemon does not automatically have access to view jobs; the user must also have read access granted at the job level to be able to view it in the Data Mover portlet. Similarly, a user who is granted read access for the individual job but not for the daemon cannot view the job in the Data Mover portlet.</p> <p>Users with write access to a job can set permissions when creating or editing the job or performing changes to job permissions for multiple jobs. Users can change permissions for multiple jobs by selecting Change Permissions in the Saved Jobs view using the Table Actions arrow in the Data Mover portlet.</p>

Job Settings Permissions

If security management is enabled, you can also restrict access to certain job settings:

- The ability to run **Update Job Steps** from the menu for a job listed in the **Data Mover** portlet.
- The maximum number of streams that a user can specify for a job.
- The utilities available. Users can be restricted to use Teradata DSA, or Teradata PT API load, update, or stream. JDBC is always available.

You access job settings permissions by clicking the **Advanced** button in the **Permissions Details** screen.

Adding Permissions

You can select users and roles to have complete access (read, write, and run) to the daemon. You can also enable security management for finer control, to designate specific permissions at both the daemon and the job level.


The following is true if security management is not enabled:


- All jobs can be viewed and run by the selected Viewpoint users in the **Data Mover** portlet.
- Individual users or roles cannot be limited to individual permissions (such as read and execute, but not write). Every user and role has all permissions (read, write, and execute) if they are selected as described here, or no permissions at all if they are not selected.

If you enable security management and set permissions at the daemon level, you can also designate that users and roles be allowed to use commands without requiring a user ID.

1. In the **Daemons** column, click the name of the daemon you want to configure.
2. In the **Daemon Setup** column, click **Permissions**.
3. Select one of the following options:


Option	Description
Give full daemon access to all selected users and roles	Security management is disabled.
Allow daemon write or execute access to be configured per user or role	Permissions are given at the daemon level.
Allow user and role access to be further restricted per job	Permissions are given at the job level.





4. [Optional] If you are setting permissions at the daemon level and want to allow command use without requiring user IDs, select the **Allow command use without user ID** check box.
5. [Optional] To filter the list of **Available Users**, use the **Available Users Filter**.
By default, the list displays a complete list of Teradata Viewpoint users.
6. [Optional] To designate permissions for users, select the user names from the **Available Users** list and click  to move them to the **Selected Users** list.
By default, a selected user has *read* access, designated by the letter **R** that is displayed near their name.

7. [Optional] To grant write and execute privileges to a user, click the user and click **Write** to grant write access and **Execute** to grant execution rights.
The letter **W** indicates that write access has been granted, and **X** indicates that execution rights have been granted.
8. [Optional] To designate permissions for a role, select the role from the **Available Roles** list and click  to move it to the **Selected Roles** list.
By default, a selected role has *read* access, designated by the letter **R** that is displayed near their name.
9. [Optional] To grant write and execute privileges to a role, click the role and click **Write** to grant write access and **Execute** to grant execution rights.
The letter **W** indicates that write access has been granted, and **X** indicates that execution rights have been granted.
10. Click **Apply**.
11. [Optional] To set additional features when security management is enabled, click **Advanced**, and select the user or role and the feature they can access.
 - a. To enable the user or role to update job plans, select the **Update job plan** check box.
 - b. To enable the user or role to change job priorities, select the **Change job priority** check box.
 - c. To set the maximum number of streams permitted, enter the number in the **Maximum number of streams** field.
 - d. To designate the utilities the user or role can use in creating jobs, select the check boxes under **Available utilities**.
12. Click **Apply**.

Configuring Event Table Logging

Teradata Ecosystem Manager events can be logged to an internal event table that Data Mover manages. You can set up and delete event tables using the **Data Mover Setup** portlet.

1. In the **Daemons** column, click the name of the daemon you want to update.
2. In the **Daemon Setup** column, click **Event Table Logging**.
3. [Optional] If there are no event tables, click **Add Event Table**, or to add additional event tables, click  next to **Edit**.
 - a. In the **Connect to System** dialog box, enter the logon information.
 - b. Click **Next**.
The **Select a Database** dialog box appears.
 - c. Next to **Select a Database**, click ▼.
You can use the filter to display only the items that match the characters you enter.
 - d. Select **OK**.
The newly created event table system and database appear.
4. [Optional] To designate the event table as the default, select the **Default** check box.
If you do not specify any event tables for logging events, the default event tables are used for jobs.
5. [Optional] To edit the username, password, or authentication mechanism, click **Edit**.

6. [Optional] To add another event table, click .
 7. [Optional] To remove an event table, click .
 8. Click **Apply**.
If the operation is successful,  appears. If the operation fails,  appears. Verify the settings are correct and try again.
- When the operation is successful, an event table is created if one does not exist under the database.





Configuring Target Databases

You can configure and manage default target databases.

1. In the **Daemons** column, click the name of the daemon you want to configure.
2. In the **Daemon Setup** column, click **Target Databases**.
3. Click **Add Target Database**.
The **Connect to System** dialog box appears.
4. In the **Connect to System** dialog box, enter the logon information.

Note:





Spaces in the user name for the source or target id causes the job to fail.

5. Click **Next**.
The **Select a Database** dialog box appears.
6. Click ▼ next to **Select a Database**.
You can use the filter to display only the items that match the characters you enter.
7. Select **OK**.
The **System** and **Target Database** display.
8. [Optional] Click  to add another target database.
9. [Optional] Click  to remove a target database.
10. Click **Apply**.
If the operation is successful,  appears. If the operation fails,  appears. Verify the settings are correct and try again.

Configuring Staging Databases

You can configure and manage default staging databases. You can configure separate staging databases for tables and for temporary artifacts, including macros, error, log, and work tables.

1. In the **Daemons** column, select the name of the daemon you want to configure.
2. In the **Daemon Setup** column, select **Staging Databases**.
3. Under **Source Staging Database For Tables**, select **Add Staging Database**.
The **Connect to System** dialog box appears.
4. In the **Connect to System** dialog box, enter the logon information.

5. Click **Next**.
The **Select a Database** dialog box appears.
6. Select ▼ next to **Select a Database**.
You can use the filter to display only the items that match the characters you enter.
7. Select **OK**.
The system and staging database display.
8. [Optional] Select  to add another staging database.
9. [Optional] Select  to remove a staging database.
10. Under **Target Staging Database For Tables** and **Target Staging Database For Temporary Artifacts**, select **Add Staging Database** and repeat steps 4 through 9 for each.
11. Select **Apply**.
If the operation is successful,  appears. If the operation fails,  appears. Verify the settings are correct and try again.

Configuring Job Timeout

Data Mover can detect and stop jobs that become unresponsive by setting user-defined timeout periods. You can specify standard timeout periods for the initiate, build, and apply phases for different object size categories.

You designate object size categories by entering sizes for *small* and *large* objects, and Data Mover uses the range between them to designate *medium* size objects. When a job is run, Data Mover identifies the largest object in the job and uses the designated timeout periods that correspond to the size of that object.

1. In the **Daemons** column, click the name of the daemon you want to configure.
2. In the **Daemon Setup** column, click **Job Timeout**.
3. Under **Settings for Job Timeout**, select the **Enable job timeout** check box to detect and stop unresponsive jobs.
4. In the **Check Frequency** field, enter the time increment in hours to check for unresponsive jobs.
5. In the **Acquisition** field under **During Data Transfer**, enter the timeout period for a job in the data transfer phase.
6. Designate the size and specify the phase timeout periods for small, medium, and large objects:
 - a. Under **Small Object Settings**, designate the size of a small object by using the **Less than** field and unit list, and enter timeout periods for the **Initiate**, **Build**, and **Apply** phases.
 - b. Under **Medium Object Settings**, enter timeout periods for the **Initiate**, **Build**, and **Apply** phases.
 - c. Under **Large Object Settings**, designate the size of a large object by using the **Greater than** field and unit list, and enter timeout periods for the **Initiate**, **Build**, and **Apply** phases.
The values you entered in the **Less than** and **Greater than** fields are automatically entered in the **From** and **To** fields under **Medium Object Settings**.
7. Click **Apply**.

Configuring Job Retry

Data Mover jobs may fail to complete successfully if they are blocked because there are locks on source or target database objects. In addition, SQL execution may fail if a transaction is aborted due to a deadlock (Error 2631). You can configure the daemon to detect these situations and attempt to start the job or deadlocked transaction after time intervals you specify.

1. In the **Daemons** column, click the name of the daemon you want to configure.
2. In the **Daemon Setup** column, click **Job Retry**.
3. [Optional] To enable blocked job detection, click the **Enable blocked jobs** check box under **Blocked Jobs**.
 - a. Next to **Attempt to start a blocked job every**, designate the time interval for **Data Mover** to wait before checking if locks are still present.
 - b. Enter the **Maximum number of blocked jobs** that can be checked for periodic restart attempts. If a job is detected as blocked when this number is reached, the job is added to the job queue.
4. [Optional] To enable deadlock error detection, click the **Enable deadlocked transactions** check box under **Deadlocked Transactions**.
 - a. Specify the time to wait before retrying a SQL execution that failed with a 2631 deadlock error in the **Time between retry attempts** fields.
 - b. Enter the **Maximum number of retry attempts** that can be checked for periodic restart attempts. If a job is detected as blocked when this number is reached, the job is added to the job queue.
5. Click **Apply**.


Configuring ID Pools





You can configure ID pools for Teradata Database source and target systems to allow job changes at a global level. Each pool has a unique name and defines one or more users for each system. When creating a job, you specify the ID pool by name; Data Mover examines the pool for the source and target system names and user profiles, and randomly picks an available user for that job.

In addition to being able to pick from a pool of users for the specified systems, ID pools enable you to change passwords in one place for all users specified in the pool.

Note:


Changing ID pool definitions can affect previously configured jobs.

1. In the **Daemons** column, click the name of the daemon you want to configure.
 2. In the **Daemon Setup** column, click **ID Pools**.
 3. In the **ID Pools** column, click  to add an ID pool.
 4. Enter an **ID Pool Name**.
 5. Click **Add System Credentials**.
 6. Select the **System** from the list.
 7. Enter the **Username** and **Password**.
-

8. [Optional] Click **Test** to verify that the login settings are correct.
If the operation is successful,  appears. If the operation fails,  appears. If you receive an error, verify that the login credentials are valid and the host can be reached.
9. Select **OK**.
10. [Optional] Click  to add an additional system.
11. [Optional] Click  to remove a system.
12. [Optional] Click **Edit** to change a selected system or its associated credentials.
13. Click **Apply**.

Configuring Cloud Staging Area

You need to configure a cloud staging area to use the Cloud Staging Copy Service.

1. In the **Daemons** column, click the name of the daemon you want to configure.
2. In the **Daemon Setup** column, click **Cloud Staging Area**.
3. In the **Cloud Staging Area** column, click  to add a cloud staging area.
4. Enter a cloud staging area **Name**.
5. Enter a **Storage Type** (Supports only S3).
6. [Optional] Select **Create Target Group** if you want Data Mover to create the required configuration on DSC.

Note:



Skip this step if you are going to use pre-existing DSC configuration.

- a. Enter AWS **Access Key ID**.
- b. Enter AWS **Secret Access Key**.
- c. Click **Add Bucket**.
- d. Enter the S3 bucket name in **Bucket**.
- e. Enter AWS **region**.
For Example: us-west-1
- f. Enter **Prefix Name**.
- g. Enter the number of **Storage Device**.
For Example: 100
7. Click **Add Source Target**.
8. Enter the **Source System** and **Target System** pairs.

Note:




Atleast one source-target pair is mandatory.

9. [Optional] If you have not selected **Create Target Group**, enter the following:

- a. Enter the **Source Target-Group** that is linked to the Source system in DSC. Make sure, the value exists as one of the target groups in DSC.
- b. Enter the **Target Target-Group** that is linked to the Target System in DSC. Make sure, the value exists as one of the target groups in DSC.
10. [Optional] Click  to add an additional source-target pair.
11. [Optional] Click  to remove a source-target pair.
12. Click **Apply**.

Configuring Advanced Settings

You can configure advanced settings to improve performance and restrict job option choices for **Data Mover** portlet users. You can configure the maximum load slots and target system restrictions for a daemon.

1. In the **Daemons** column, click the name of the daemon you want to configure.
2. In the **Daemon Setup** column, click **Advanced**.
3. Enter the **Maximum QueryGrid T2T Tasks** to designate how many QueryGrid T2T tasks can be executed at the same time.
The default value is read from the daemon. The permitted range is 0 to 1000 inclusive.
4. [Optional] To enable QueryGrid 2.0 or later to wait for accurate byte counts and report job status, select the **For QueryGrid v2.x and higher tasks, wait for accurate byte counts after copying data (will delay job finish)** check box.
This option is not enabled by default.
5. [Optional] To allow users to specify different character sets when copying data in a job, select the **Support different character set for copying** check box.
6. [Optional] To change how many Teradata PT API load or update jobs can be run against target systems at the same time from the default value, do the following under **Maximum Load Slots**:
 - a. Enter the **Default Maximum Load Slots**, and click **Add Load Slot Limit**.
The default value is read from the daemon. The permitted range is 1 to 99999 inclusive.
 - b. Select the **System** from the list and enter the **Maximum Slots**.
The number defaults to the system default. The permitted range is 1 to 99999 inclusive.
 - c. [Optional] Click  to designate the maximum slots available for another system.
 - d. [Optional] Click  to remove maximum slots designation from a system.
7. [Optional] To prevent a system from being available for selection as a target system, select it from the **Valid targets** box under **Target System Restrictions** and click the  button to move it to the **Never a Target** list.
An error occurs if a Data Mover user tries to designate an excluded system as a target.
8. Do not use **Additional Users for Target System** unless you are upgrading from an older version and were using this feature.
Use ID Pools instead.
9. [Optional] To define the source and target systems in a system pair, do the following:

- a. Under **Force Direction Pairs**, click **Add Pair** and select the systems from the **Source** and **Target** lists.
- b. To allow any statistics in the target database to be returned to the source, select the **Allow Stats Back** check box.
- c. [Optional] Click ☐ to add another system pair.
- d. [Optional] Click ☐ to remove a system pair.

Users cannot switch the direction after systems are paired as source and target databases.

10. Click **Apply**.

Data Mover

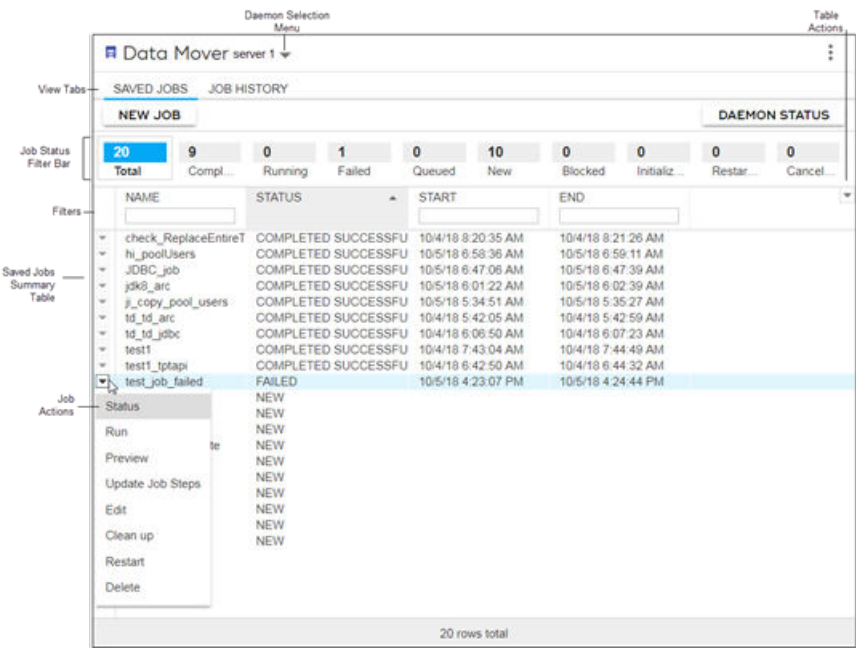
The **Data Mover** portlet is a Java-based interface that allows flexible, asynchronous copies of database objects in a high-availability environment. It is useful for incremental data copy between Teradata Database systems as well as quick full-table copies. A high-availability multi-system pair can copy data from one system to another and execute transactions for database objects on both systems.

The **Data Mover** portlet allows the Teradata Database Administrator to:

- Establish systems to enable copying, or recover systems to re-enable copying.
- Copy a large volume of data from one Teradata Database system to another on a regularly-scheduled basis.
- Perform a partial copy of tables from a source Teradata Database system to a target system.

The Saved Jobs View

The **Saved Jobs** view in the **Data Mover** portlet enables you to select a daemon and display newly created and the most recently run jobs. It also enables you to create new jobs and perform key actions, such as running and editing saved jobs. No jobs are listed if you have not previously created and saved a job. The list of saved jobs is updated every 60 seconds.



View Tabs

The **Saved Jobs** view displays all created and saved jobs, and the **Job History** view displays all instances of job executions.

New Job Button

Creates a new Data Mover job.


Job Status Filter Bar

Provides a count of the jobs in each of the statuses. Click a number to display the jobs by job status, or select additional statuses from the list.


Filters

Shows only rows that match your filter criteria. Filter criteria include the job name defined when the job was created, as well as the job starting or ending timestamps that Data Mover adds when running a job.

Saved Jobs Summary Table

Lists the job name, job status, and the start and end times of the job. The first column displays a Job Action icon  for each job.

Job Actions

Provides a  list of actions available for each job, depending on the status of the job. The actions that appear in the list include **Run**, **Delete**, **Restart**, **Clean Up**, **Status**, **Preview**, **Edit**, and **Change Priority**. Some actions may be disabled, depending on the permissions available to you.

Daemon Selection Menu

Enables you to select the daemon whose activity you want to monitor.

Daemon Status Button

Lists the status of agents and tasks. When tasks are queued, an information icon appears next to this button.

Table Actions

Enables you to configure the columns that display and to clear the filters applied to the table. Provides options to change permissions, change priority, and update job steps for multiple jobs.

About the Job Status Filter Bar

The job status filter bar allows you to filter on a specific job status in the **Saved Jobs** or **Job History** view.



The job status filter bar buttons provide a count of jobs in each status category. Click any status button to filter for the selected job status or select a job status from the overflow menu. For example, click **Completed** to display all jobs that have successfully completed.

In the **Saved Jobs** view, you can select a job status from the **Overflow Menu** to replace a job status currently showing on the job status filter bar. The **Job History** view displays only the statuses for jobs that have been executed. The possible job statuses are as follows:

Total

All saved jobs (in the **Saved Jobs** view) or all job executions (in the **Job History** view)

Completed

Jobs that have run to completion

Running

Jobs currently in progress

Failed

Jobs that have failed to run to completion

Queued

Jobs that are waiting for resources to become available before they can begin running

New

Jobs that have never been run

Blocked

Jobs currently blocked due to locks on source or target objects

Initializing

Jobs in the process of initializing

Restarting

Jobs in the process of restarting

Cancelled

Jobs cancelled after they were started

Running a Job

You can run any job in the **Saved Jobs** view, regardless of the current status of the job.


1. Select ☐ next to a job.
2. Click **Run**.
The **Job Status** view appears and refreshes with information about the currently running job.

Previewing a Job

Before running a saved job, you can preview the job description and other job settings.

1. Select ☐ next to a job.
2. Select **Preview**.
The **Preview Job** view appears.

Note:

If there is a lengthy list of objects, select ☐ in any of the filter boxes at the top of the **Preview Job** view to sort by that column. You can also type any portion of an object in the filter boxes to locate a particular object by that column of information. Select  in any filter box for a list of wildcards available for that particular column.

3. Select **View Steps** to see a list of the job steps.
Some job steps display a **View Details** link that enable you to view scripts.
4. Select **View Streams/Sessions** to view stream and session details of the job.

Editing a Job

You can edit a saved job that appears in the **Saved Jobs** view to change the job definition, credentials, or other job settings.

Each job definition is associated with a job name. Each time you run a job using a particular job definition, you create an instance of the job with that name. If you run a job more than one time, you create multiple instances of the job based on that job definition. When you edit a job, you edit the underlying job definition, as well as all the instances of the job based on that job definition.

1. Select ☐ next to a job.
2. Click **Edit**.

If you are the job owner (the user who created or last edited the job), the **Objects** tab of the **Saved Jobs** view appears. If you are not the job owner, you must enter credentials for the source and target systems.

Note:

Spaces in the user name for the source or target id causes the job to fail.

Note:

Spaces in the account name for the source or target account id causes the job to fail.

3. [Optional] Provide a new name for the job.
Naming considerations:
 - If you do not rename the job, any job edits you make overwrite the existing job definition when you click **Save**.
 - If you give the job a different, unique name, any job edits you make are used as a new job definition when you click **Save**. Thus renaming a job is equivalent to creating a new job.
 - If you give the job an existing job name and click **Save**, an error results.
4. Modify the information in the **Objects** and **Job Settings** tabs as desired.
5. Click **Save**.
If any database objects in the original job no longer exist, a message displays. Click **OK** and click **Save** again.


Changing Job Permissions



When you create or edit a job, you can set permissions to allow designated users and roles read, write, and execute access. Afterwards you can change permissions for saved jobs.

The privileges associated with job permissions are as follows:

Access Type	Privileges
Read (R)	Preview jobs View status of running or executed jobs
Write (W)	Edit jobs Delete jobs Update job steps Change Job Priority Edit job scripts
Execute (X)	Run jobs Stop jobs Restart jobs Clean up jobs

In order for these permissions to have any effect, the Teradata Viewpoint Administrator must first enable security management in the **Data Mover Setup** portlet.

1. From the **Saved Jobs** view, click  in the table header, and select **Change Permissions**.
2. Select the saved jobs for which you want to change permissions, and click **Next**.
3. Select users and roles to grant access.

Option	Description
Users	<ol style="list-style-type: none"> a. Select one or more Viewpoint users and click  to move them to the Users with access box. An R appears to indicate that they have read access. b. Use the Write and Execute buttons to grant additional write and execute privileges.
Roles	<ol style="list-style-type: none"> a. Select one or more Viewpoint roles and click  to move them to the Roles with access box. b. Use the Write and Execute button to grant additional write and execute privileges.

4. Select **OK**.

Changing Job Priority

After a daemon reaches **MaxConcurrentJobs** status, any new job is placed in a queued status. You can update the job priority from the **Saved Jobs** view to change the execution order in the queue. For example, if jobs A and B are waiting in the queue and you want to run job B first, set the priority for B to **High** and the priority for A to **Medium**.

Note:

To modify the priority of jobs in the queue, the user or role must have **Change job priority** permissions set up in the **Data Mover Setup** portlet.

1. Select ☐ next to a job.
2. Select **Change Priority**, and then select **Low**, **Medium**, or **High**.
A confirmation message appears.
3. Select **OK**.

Changing Priority for Multiple Jobs

After a daemon reaches **MaxConcurrentJobs** status, any new job is placed in a queued status. You can update the job priority from the **Saved Jobs** view to change the execution order in the queue. For example, if jobs A, B, C, and D are waiting in the queue and you want to run C and D first, then set the priority for C and D to **High**.

Note:

To modify the priority of jobs in the queue, the user or role must have **Change job priority** permissions set up in the **Data Mover Setup** portlet.

1. Click ☐ in the table header and select **Change Priority**.
2. Select the check boxes for the jobs whose priority you want to change.
You must select at least one job to perform the priority update action.
3. Click **Next**.
4. Select the desired priority (**High**, **Medium**, or **Low**).
5. Click **OK**.

Updating Job Steps for a Single Job

When you created or edited a job, you may have selected the **Freeze job steps** check box to prevent Data Mover from verifying whether there have been any changes in source and target environments since the job was created.

You can update the job steps for a job that has frozen steps. For example, if any changes have occurred in the source or target environment, you can update the job steps so they are refreshed.

1. In the **Saved Jobs** summary table, click ☐ next to the name of the job.
2. Click **Update Job Steps**.
A confirmation message appears.
3. Click **OK** to return to the **Saved Jobs** view.

Updating Steps for Multiple Jobs

When you created or edited a job, you may have selected the **Freeze job steps** check box to prevent Data Mover from verifying whether there have been any changes in source and target environments since the job was created.

If there are multiple jobs with frozen steps, you may want to refresh and update the steps, particularly if there have been some changes in the source or target environments from the time the jobs were created. You can select one or more jobs to refresh their job steps.

1. From the **Saved Jobs** view, click ☐ in the table header, and select **Update Job Steps**.
A list of jobs whose steps are currently frozen displays.
2. [Optional] To locate the jobs you are looking for, use the filters as follows:
 - In the **Name** and **Status** fields, enter text.
 - Click in the **Start** or **End** fields, click the menu, and select **In the last**, **After**, or **Before**, select the timestamp cutoff duration, and click **OK**. If you select **Range**, specify the range date and time, and click **OK**.
3. Select the check boxes for the jobs whose steps you want to update.
You must select at least one job to perform the step update action.
4. Click **Update**.
5. Click **Update job steps** to confirm.
A confirmation message appears.
6. Click **OK** to return to the **Saved Jobs** view.

Viewing Job Steps

You can examine the sequence of steps that **Data Mover** performs when a saved job is executed.

1. Select ☐ next to a job.
2. Click **Preview**.
3. Click **View Steps**.
4. [Optional] Click **View Details** next to a job step for more information about the step.

Viewing Streams and Sessions

For Teradata to Teradata jobs, you can examine the system-generated and user-defined values that will be used the next time a job is run.



1. Select ☐ next to a job.
2. Click **Preview**.
3. Click **View Streams/Sessions**.

Note:

To view the values used when the job ran previously, see [Viewing Job Status](#).

Viewing Job Status

You can view the job status of the most recently run job from the **Saved Jobs** view. You can view the job status of any previously job run from the **Job History** view.



- To view the job status of the most recently run job:
 1. In the **Saved Jobs** view, click  next to the name of the job.
 2. Click **Status**.
- To view the status of a previous run job:
 1. In the **Job History** view, click  next to the name of the job.
 2. Click **Status**.

The **Job Status** view appears and displays the following information.

Field	Description
Job Name	Identifies the name of the job.
Source	Identifies the source Teradata Database system.
Target	Identifies the target Teradata Database system.
Start	Provides the time the job began to run.
End	Provides the time the job stopped running. If the job is currently running, two hash lines display.
Size	Displays the total job size in KB, MB, or GB. The size is an estimate until the job completes running; the completed size may vary from the number shown during the initial phase.
Job Steps	Indicates Frozen if the job steps have been set to be frozen in the Job Settings tab.
Progress Bar label	Displays the current status (Running, Completed, Cancelled, Failed, or Blocked) and the percentage of the job completed. Continually updated for running jobs.
Elapsed Time	Displays the run time for the job. Continually updated for running jobs.
Objects	Displays the number of objects copied.
Copied	Identifies the size of data transferred in KB, MB or GB. Continually updated for running jobs.
Stop button	Provides the ability to cancel a running job.
Streams tab	Displays data streams used between the source and target databases. Enables you to export the data streams into a .csv file. Data stream details can also be viewed in the Log tab.
Log tab	Displays a detailed log that the Teradata utility and Data Mover generate while processing the job, including a table summarizing the start time, type, duration time, and message for each job event. Enables you to export the data stream detail information into a .csv file.


Viewing Job Logs

You can see detailed information about a job by viewing its log. You set or modify the level of detail that displays using the **Logging level** field in the **Job Settings** tab for the job.

1. Click  next to the job whose log you want to view.
2. Click **Status**.
3. Click the **Log** tab.
4. [Optional] To export the information to a .csv file, click .


Restarting a Job

You can only restart a failed job, cancelled job, or a job that you stop before it completes.

1. Click **Failed** in the job status filter bar to filter the **Saved Jobs** summary table.
2. Select  next to a job.
3. Select **Restart**.
The **Job Status** view appears and refreshes with information about the job you are restarting.

Cleaning Up a Job

The utilities TPT API, Teradata JDBC, and Teradata DSA generate temporary tables and place locks when running a job. If a job fails or is stopped before successful completion, you can remove these job artifacts. You can only clean up failed or incomplete jobs.

1. Select  next to a job.
2. Click **Clean Up**.

Deleting a Job

Each job definition is associated with a job name. Each time a job is run, a job instance is created. If a job is run multiple times, multiple instances of the job based on that job definition are created. The latest instance is listed in the **Saved Jobs** view, and any earlier instances are listed in the **Job History** view.

You can delete a job instance from the **Saved Jobs** or **Job History** view so that the job is no longer listed and you can no longer view the job history. When you delete a job instance, the underlying job definition and all other job instances based on it are still available and are still listed.

From the **Saved Jobs** view, in addition to deleting the most recent run jobs, you can choose to delete all instances of the job based on the job definition, or delete the job definition itself and all instances. When you delete a job definition and all instances:

- The underlying job definition and history of the job instances are removed.
- All jobs instances based on the job definition are deleted.

- The job is removed from the **Saved Jobs** and **Job History** views.

You cannot delete a job that is in Running, Queued, or Blocked status.

- In the **Saved Jobs** or **Job History** view, click ☐ next to the job.
- Click **Delete**.
- For a job in the **Job History** summary table, click **OK** to confirm the deletion. For a job in the **Saved Jobs** summary table, do one of the following:

Delete Options	Description
The selected instance	<ol style="list-style-type: none"> Select Delete only this instance Click Next. Click OK to confirm that you want to remove this instance of the job history.
All instances	<ol style="list-style-type: none"> Select Delete all instances Click Next. Click OK to confirm that you want to remove all instances of the job history.
Job definition and all instances	<ol style="list-style-type: none"> Select Delete the job definition and all instances Click Next. Click OK to confirm that you want to delete the job definition and all associated instances.

Daemon Status

You can view agent and task status in the **Daemon Status** view. The **Agent Status** section lists the number of active tasks as well as the maximum tasks available.

When tasks are pending, Data Mover displays an information icon next to the Daemon Status button. Move the mouse over the icon to see the number of tasks in queue.

The **Task Status** section lists information for all tasks Data Mover is currently processing. Tasks are utility-specific and are run by the Data Mover agent. From a high-level perspective, this view enables you to identify tasks that use task-related resources in Data Mover, such as agent task slots and load slots. From a job-level perspective, it identifies work that is blocking a particular job, and the length of the delay before starting the job.

Listing Data Mover Active Tasks

- From the **Saved Jobs** view, click **Daemon Status**.

2. [Optional] Click a task under **Task Status** to display **Job Status** information for the task.

Related Information:

[list_tasks Command Reference](#)

The Job History View

The **Job History** view in the **Data Mover** portlet enables you to select a daemon and display a list of all instances of jobs that have been run. You can view the status of and delete job instances.

Labels in the screenshot:

- Daemon Selection Menu
- Table Actions
- Data Mover server 1
- SAVED JOBS
- JOB HISTORY
- 10 Total
- 9 Completed
- 1 Failed
- 0 Cancelled
- NAME
- STATUS
- START
- END
- test_job_failed
- test1_tptapi
- Status
- Delete
- jdbbc
- _arc
- _y_pool_users
- jdk8_arc
- hi_poolUsers
- check_ReplaceEntire
- JDBC_job
- 10 rows total
- Jobs History Summary Table
- Job Actions

NAME	STATUS	START	END
test_job_failed	FAILED	10/5/18 4:23:07 PM	10/5/18 4:24:44 PM
test1_tptapi	COMPLETED SUCC	10/4/18 6:42:50 AM	10/4/18 6:44:32 AM
jdbbc	COMPLETED SUCC	10/4/18 7:43:04 AM	10/4/18 7:44:49 AM
_arc	COMPLETED SUCC	10/4/18 6:06:50 AM	10/4/18 6:07:23 AM
_y_pool_users	COMPLETED SUCC	10/4/18 5:42:05 AM	10/4/18 5:42:59 AM
jdk8_arc	COMPLETED SUCC	10/5/18 5:34:51 AM	10/5/18 5:35:27 AM
hi_poolUsers	COMPLETED SUCC	10/5/18 6:01:22 AM	10/5/18 6:02:39 AM
check_ReplaceEntire	COMPLETED SUCC	10/5/18 6:58:36 AM	10/5/18 6:59:11 AM
JDBC_job	COMPLETED SUCC	10/4/18 8:20:35 AM	10/4/18 8:21:26 AM
JDBC_job	COMPLETED SUCC	10/5/18 6:47:06 AM	10/5/18 6:47:39 AM

Job Status Filter Bar

Provides a count of the jobs in each of the statuses. Click a number to display the jobs by job status, or select additional statuses from the list.

Filters

Shows only rows that match your filter criteria. Filter criteria include the job name defined when the job was created, as well as the job starting or ending timestamps that Data Mover adds when running a job.

Job History Summary Table

Lists a summary of the job name, job status, and the start and end times for each job instance.

Job Actions



Provides a  list of actions available for each job. You can select a job and view its status or delete it.

Table Actions

Enables you to configure the columns that display and to clear the filters applied to the table. Also provides options that enable you to change permissions and update job steps for multiple jobs.

Viewing Job History

You can view a list of all Data Mover jobs that have been executed, regardless of whether they completed successfully.

1. Click the **Job History** tab in the **Data Mover** portlet.
The list of jobs displays.
2. To view more details about a particular job execution, click , and click **Status**.


About Planning a Job





When you create a job in the **Data Mover** portlet, you select the source and target databases to use and specify the source database objects to copy. In addition, the portlet provides numerous settings to refine the job definition and to control performance, logging, and other factors. You may want to consider the following when planning a job:

- **Favorite Systems:** You can provide the logon information for target and source systems you use often and have the settings used as defaults, so that you do not need to enter the information each time you create a job.
- **Object copy settings:** Depending on the type of object type you are copying, there are various copy settings you can designate, such as a staging database to use if the target does not have enough space, the key columns for a table, whether to compare DDL before copying, and whether to override lock access.
- **Job settings:** There are default settings that are applied upon job creation, but you can change the defaults for a number of parameters, including job priority, logging level, job permissions, event table logging, and overwrite behavior if the objects already exist on the target.

Setting Favorite Systems

You can specify credentials and settings for favorite source and target Teradata Database systems so you do not need to enter that information each time you choose those databases.

1. Click  in the portlet frame and select **Settings**.
2. Specify a system.

3. Enter authentication information for connecting to the system.
4. Select the **Source** or **Target** check box for the system.
You can specify a system as source and target by selecting both check boxes.
5. [Optional] Click  to add an additional system.
6. [Optional] Click  to remove a system.
7. [Optional] Click **Test** to verify that the login settings are correct.
If the operation is successful,  appears. If the operation fails,  appears. If you receive an error, verify that the login credentials are valid and the host can be reached.
8. [Optional] To use these settings when new instances of the **Data Mover** portlet are added, click **Save as Default**.
9. [Optional] To reset the user-defined default settings to system values when new instances of the **Data Mover** portlet are added, click **Clear Defaults**.
10. Select **OK**.

Related Information:

[System Credentials](#)

Creating a Job

Before creating a job, you may want to add source and target systems to the **Favorite Systems** tab of the **Settings** view.

Note:

Spaces in the user name for the source or target id causes the job to fail.

Note:

Spaces in the account name for the source or target account id causes the job to fail.

1. From the **Saved Jobs** view, click **New Job**.
2. Enter a unique name for the job.
3. Edit the credentials for the **Source** and **Target** systems.
4. Select objects from the source system in the **Objects** tab.
5. [Optional] To verify the parent database and objects selected, click the **Selection Summary** tab.
6. [Optional] To adjust job settings for the job, click the **Job Settings** tab.
Settings can include specifying whether a job continues or aborts if an access rights violation is encountered on an object.
7. Click **Save**.

The newly created job is listed in the **Saved Jobs** view.

Related Information:

[Setting Favorite Systems](#)

[Editing System Credentials](#)

[About the Job Summary](#)

[About Job Settings](#)

System Credentials

Teradata System Credentials

Parameter	Description
Nickname	Name of the Teradata Database system.
Username	Teradata Database user ID.
Password	Teradata Database password.
Account String	Characteristics for profile member sessions in Teradata Database. An account string identifies which account is charged for the space used by the user and session and may include a priority-level Performance Group prefix code which establishes the session priority.
Authentication	Method of authenticating the user and establishing a session based on user privileges. Options might include TD2 or LDAP. If an authentication mechanism is not selected, the system uses the default authentication mechanism for the Teradata Database.
Character Set	Character set the system uses to communicate with Teradata Database. If not defined, Teradata Viewpoint uses Teradata Database system settings.




Editing System Credentials

When you create a job, the system credentials for the source and target systems must be specified.

1. In the **New Job** dialog box, click **Edit** next to the Source or Target field.
2. Select the **System Type**.
3. If the target or source system type is a Teradata system, select the **Credential Type**, and provide the additional information applicable to your choice.

Option	Description
Free Form Credentials	<p>If no ID pools have been created, you must enter all required credential information manually.</p> <ol style="list-style-type: none"> a. Select a System from the list. If you have created Favorite Systems, you can click Autofill Favorite System, select the system, and click Load so that credential information is filled in automatically. b. Enter a Username and Password.

Option	Description
	c. [Optional] Enter the Account String . d. Select the Authentication Mechanism . e. Select the Character Set .
ID Pool	If ID pools have been created, you can select an existing ID pool. The source and target systems must use the same ID pool. a. Select a System from the list. b. Select an ID Pool from the list c. Select the Character Set .

4. [Optional] To verify that the settings are correct, click **Test**. If the operation is successful,  appears. If the operation fails,  appears. Verify the settings are correct and try again.
5. [Optional] To add the system to your favorites or overwrite information previously entered for a favorite system, click **Add to Favorites**.
If you want to change information previously entered, click **Overwrite**. If the operation is successful,  appears.
6. Click **OK**.

About Copy Settings


When you select an object in the object browser, you can use a dialog box associated with it to designate optional copy settings. The specific copy settings available depend on the type of object you select.

The settings you can designate include:

- An alternative location on the target to which you want to map the object. For example, you can copy a user or database from the source system to a different database in the target system.
- A staging database. For example, if the target database does not have enough space, specifying a staging database creates the tables in the staging database instead of in the target database.
- A separate staging database for temporary artifacts. For example, if you are copying a table, you might want to designate that any macros or work, log, or error tables be copied to a staging database other than that where the staging table is held.
- Whether data, statistics, or both should be copied.
- Key columns, and whether to validate row count for a table.
- Whether to compare the DDL of source and target table before copying.
- Staging to target mechanism.
- Spooling, if the Teradata PT API utility is used.
- Whether to override lock access if the Teradata PT API or JDBC utilities are used.
- Whether and when to activate a trigger.

Copying a Database

You can copy database objects to the same or to a different location on a target Teradata system. You can choose a staging database for tables and a separate staging base for temporary artifacts that are created, such as work, log, or error tables, or macros. You can also choose to replace an entire database by eliminating all of the objects in the target database and replacing them with those from the source database.

1. In the **Objects** tab, click  to expand the tree.
2. Click ▾ to the right of the database name to display the **Database Settings** dialog box for the object. The dialog box opens, with the name of the database displayed at the top.
3. [Optional] To **Map to a different database on target**, click ▾ and select a database. You can use the filter to display only the items that match the characters you enter.
4. [Optional] To set a **Source staging database for tables**, click ▾ and select a source database. You can use the filter to display only the items that match the characters you enter.
5. [Optional] To set a **Target staging database for tables**, click ▾ and select a target database. You can use the filter to display only the items that match the characters you enter.
6. [Optional] To set a **Target staging database for temporary artifacts**, click ▾ and select a target database. You can use the filter to display only the items that match the characters you enter.
7. [Optional] To verify that important aspects of the DDL for the source and target tables match before performing the copy, select an option from the **Compare DDL** list.

Option	Description
Default	By default, Data Mover compares the DDL.
True	Compare the DDL before performing the copy.
False	Do not compare the DDL before performing the copy.

8. [Optional] To eliminate all objects in the target database that do not exist in the source database, click the **Replace entire target database** check box. Selecting this check box removes any additional tables that existed on the target database, but not on the source database.
9. Select **OK**.

Copying a Table

You can copy table data, statistics, or both to the same or a different location on a target system. You can rename tables, validate row count, perform partial table copy, and designate additional table copy options.

Tables can be copied between systems supported by Data Mover.

1. In the object browser, click the check box next to the name of the table you want to copy.
2. Click ▾ to display the **Table Settings** dialog box.
The name of the table displays at the top of the dialog box.
3. [Optional] For Teradata tables, select one of the following criteria under **Copy the following** to restrict the copy job.
 - Only data
 - Only statistics
 - Both data and statistics
4. [Optional] Enter the predicate of the WHERE restrictive clause in the **Include Only Where** box to perform a partial table copy.
Do not enter the keyword WHERE. For example, if you want to restrict a partial table copy to only the columns in which col1 is greater than 4, type `col1 > 4` in the box.
5. [Optional] Enter the name in the **Rename to** box to use an alternate name for the table on the target database.
6. [Optional] To map to a different database or designate staging tables for tables and temporary artifacts, click ▾ next to the associated field, and select a database.
You can use the filter to display only the items that match the characters you enter.
7. [Optional] Select the **Use source staging table** check box to enable source staging.
8. [Optional] If **Use source staging table** is enabled, select a source database from **Source staging database for tables**.
If a database is not specified, Data Mover uses the same database as the table being copied. Data Mover automatically names the staging table.
9. [Optional] To use a separate staging database for the target, click ▾ to the right of **Target staging database for tables** and select a database.
10. [Optional] Select the **Use target staging table when table exists on target** check box to force a target staging table to be created.
11. [Optional] To use a separate staging database for temporary artifacts, click ▾ next to **Target staging database for temporary artifacts** and select a database.
12. [Optional] Select one of the following options from the **Validate Row Count** list to verify that the number of rows in the source and target tables match after completing the copy job.

Option	Description
None	Do not check row count. None is the default setting.
Partial	Perform a row-count check on the subset of data that was copied. Only applies when doing partial table copies.
All	Perform a row-count check on the entire table. Applies to full or partial table copies.

13. [Optional] Select one of the following options from the **Compare DDL** list to verify that important aspects of the DDL for the source and target tables match before performing the copy.

Option	Description
Default	By default, Data Mover compares the DDL.
True	Compare the DDL before performing the copy.
False	Do not compare the DDL before performing the copy.

14. [Optional] Select one of the following options from **Database to Client Encryption** to enable encryption at the job, table, or view level.

Option	Description
Default	Use the encryption setting specified in the daemon settings.
True	Override daemon settings and enable encryption.
False	Override daemon settings and disable encryption.

15. [Optional] Select one or more key columns from the **Key Columns** list to specify which columns uniquely identify each row when upserting them into the target table in a partial table copy. You must specify at least one key column from the list. You can select multiple columns by holding the **Ctrl** key and clicking the column names in the list. Specify all primary indexes as key columns if at least one of the primary indexes is unique.
16. [Optional] Click **Advanced**.
17. [Optional] Select one of the following options from the **Staging to Target** list to designate how to copy data from a staging table to a target table when a target table already exists.

Option	Description
Default	Data Mover chooses the most efficient way to copy data from the staging table to the target table.
Delete insert	Rows from the target table are deleted with the DELETE statement, then copied from the staging table to the target table with the INSERT/SELECT statement. The rows to be deleted depend on the job. With a partial table copy, only the rows that match the SQL query are deleted. With a full table copy, all rows are deleted.
Merge	Uses the MERGE statement to copy data. An error results if the MERGE statement cannot be used. The MERGE statement cannot be used to copy multiset tables.
Insert only	No rows are deleted from the target table. Instead, the INSERT/SELECT statement copies rows from the staging table to the target table.

18. [Optional] For Teradata to Teradata jobs only, select one of the following options from the **Export Without Spool** list to control the spooling that precedes a table copy when using the Teradata PT API utility.
- You can export a table without spooling only in Teradata Database 13.10 or later.

Option	Description
Default	By default, the Teradata Database does not write to spool space before copying tables.
True	Never write to spool space before copying the table.
False	Always write to spool space before copying the table.

19. [Optional] For Teradata to Teradata jobs only, select the **Override lock access** check box to allow updates to the source table when copying to the target.




NOTICE

Use this option with extreme care. If you select this option, you risk copying data to the target table that has not been committed on the source table. This option only applies to copy jobs that use Teradata PT API and Teradata JDBC.

20. [Optional] To select a map when copying at the object level, click ▾ to the right of **Target System Hash Map** to view a list of available maps.
When **Default** is selected, the system determines the best map for the job.
21. Select **OK**.

Copying a View

You can copy a view definition and the underlying view data to a table or a view on a target Teradata system.

- In the **Objects** tab, select  to expand the tree.
- Select  to expand a database or user object where the view is located.
- To filter so only views display, do the following:
 - Select  and clear all check marks except for **View**.
 - Move the cursor off of the object-types filter to refresh the tree with views in the sub-branch.
- Select ▾ to display the **View Settings** dialog box.
The name of the view displays at the top of the dialog box.
- [Optional] Under **Copy the following**, specify what you want to copy:
 - Only definition
 - Only data to table
 - Both definition and data to table
 - Both definition and data to view
- Do one of the following:
 - If you are copying only the definition, skip to step 13 and you are finished.
 - If you are copying data, continue with steps 7 through 12.
- [Optional] To perform a partial table copy, enter the predicate of the WHERE restrictive clause in the **Include Only Where** field.

Do not enter the keyword WHERE. For example, if you want to restrict a partial table copy to only the columns in which col1 is greater than 4, type `col1 > 4` in the field.

8. [Optional] If you are copying only data or both definition and data to a table, enter the target table name in the **Target table name** field.
9. [Optional] If you are copying only data or both definition and data to a table, select an available database from **Map to Different Database on Target**.
You can use the filter to display only the items that match the characters you enter.
10. [Optional] Select the **Use source staging table** check box to enable source staging.
This option is enabled when **Only data to table** or **Both definition and data to table** is selected in the **Copy the Following** field.
11. [Optional] If **Use source staging table** is enabled, select a source database from **Source staging database for tables**.
If a database is not specified, Data Mover uses the same database as the table being copied. Data Mover automatically names the staging table.
12. [Optional] To use a separate staging database for tables, select an available database from **Target staging database for tables**.
You can use the filter to display only the items that match the characters you enter.
13. [Optional] Select the **Use target staging table when table exists on target** check box to force a target staging table to be created.
14. [Optional] To use a separate staging database for temporary artifacts, select an available database from **Target staging database for temporary artifacts**.
You can use the filter to display only the items that match the characters you enter.
15. [Optional] To verify that the number of rows in the source and target tables match after completing the copy job, select one of the following options from the **Validate Row Count** list.

Option	Description
None	Do not check row count. None is the default setting.
Partial	Perform a row-count check on the subset of data that was copied. Only applies when doing partial table copies.
All	Perform a row-count check on the entire table. Applies to full or partial table copies.

16. [Optional] To verify that important aspects of the DDL for the source and target tables match before performing the copy, select one of the following options from the **Compare DDL** list.

Option	Description
Default	By default, Data Mover compares the DDL.
True	Compare the DDL before performing the copy.
False	Do not compare the DDL before performing the copy.








17. [Optional] Select one of the following options from **Database to Client Encryption** to enable encryption at the job, table, or view level.


Option	Description
Default	Use the encryption setting specified in the daemon settings.
True	Override daemon settings and enable encryption.
False	Override daemon settings and disable encryption.

18. [Optional] For a partial table copy, select one or more key columns from the **Key Columns** list to specify which columns uniquely identify each row when upserting them into the target table. You must specify at least one key column from the list. You can select multiple columns by holding the **Ctrl** key and selecting the column names in the list. Specify all primary indexes as key columns if at least one of the primary indexes is unique.
19. Select **OK**.

Copying Join and Hash Indexes

You can copy data, statistics, or both data and statistics for hash or join indexes to a target Teradata system. You can also give indexes a different name or copy them to a different database on the target.

- In the **Objects** tab, click  to expand the tree.
- Click  to expand a database or user object where the join or hash index is located.
- To filter so only hash indexes or join indexes display, do the following:
 - Click  and clear all check marks except for **Join Index** or **Hash Index**.
 - Move the cursor off of the object-types filter to refresh the tree with indexes in the sub-branch.
- Click  to display the index settings dialog box.
The name of the index displays at the top of the dialog box.
- [Optional] From the **Copy the following** list, select an option for the copy job:
 - Only definition
 - Only statistics
 - Both definition and statistics
- [Optional] To use an alternate name for the index on the target database, enter the name in the **Rename to** box.
- [Optional] To map to a different database on the target, click  next to **Map to a different database on target**.
You can use the filter to display only the items that match the characters you enter.
- [Optional] To select the table associated with the index:
 - Click **Browse**.
 - Click  to expand the tree.
 - Click  to expand one or more sub-branches.





- d. Click  in the sub-branch to display the object-types filter. Filter the sub-branch by selecting **Table**.
- e. Select one or more tables associated with the index.
- f. Click **Select** at the bottom of the **SELECT ASSOCIATED TABLES** view to complete the table selection.

The results of your selection appear in a summary table that lists the database and table associated with the index.

9. Select **OK**.




Copying a Trigger

You can copy triggers to a target Teradata system. If you enable the trigger, you can specify whether to place the trigger on the target table before or after the table is loaded.

1. In the **Objects** tab, click  to expand the tree.
2. Click  to expand a database or user object where the trigger is located.
3. To filter so only triggers display, do the following:
 - a. Click  and clear all check marks except for **Trigger**.
 - b. Move the cursor away from the object-types filter to refresh the tree with triggers in the sub-branch.
4. Click  to display the **Trigger Settings** dialog box.
5. [Optional] To activate the trigger and designate when to place it on the target table, select **Enabled** and do one of the following from the **Action Time** list:
 - Select **BEFORE** to place the trigger on the target table before the table is loaded. A trigger action occurs for each row that is copied from the source.
 - Select **AFTER** from the **Action Time** list to place the trigger on the target table after the table is loaded. No trigger action occurs for rows that are copied from the source.
6. Select **OK**.

Copying Foreign Server Object Definitions

The foreign server object definitions available for copying are the foreign definitions created in the Data Mover command-line interface. For information about foreign server object definitions, see [About Copying Foreign Server Objects](#).

1. In the **Objects** tab, click  to expand the tree.
2. Click  to expand a database or user object where the foreign server is located.
3. To filter so only foreign server objects display, do the following:
 - a. Click  and clear all check marks except for **Foreign Server**.
 - b. Move the cursor away from the object-types filter to refresh the tree with foreign servers in the sub-branch.
4. Select **OK**.

About the Job Summary

The **Selection Summary** tab enables you to view information about the objects to be copied to the target system, as well as any objects you rename or databases you remap, before saving the job.

Object Icon

Displays an icon of the object type.

Source DB

Provides the name of the source database.

Source Obj

Provides the name of the object in the source database.

Size

Identifies the object size in megabytes or kilobytes (K).

Rename to

Identifies the new name you give to objects copied to the target.

Map to

Identifies the target database to which you map a source database.

Note:

If you are copying a partial table, you may see the following display: Size estimate based on full table instead of partial table. This text appears when both of the following apply:

- You entered a WHERE restrictive clause in order to copy a partial table.
- You do not have access on the source system to calculate the precise size, which requires running multiple queries on the source system.

Access to calculating the precise size for partial table copying is set using portlet permissions in the **Portlets** tab of the **Roles Manager** portlet.

About Job Settings

Jobs have default settings that are applied after a job is created. You can make changes to the default job settings using the **Job Settings** tab.

Field	Description
Priority	Set the relative priority for starting a job. Data Mover uses this priority to determine which jobs to run first when multiple jobs are in the execution queue. This priority is also used to determine which tasks are run first. The options are Low, Medium, and High.

Field	Description
	The default is Medium. Jobs designated as high priority are started before lower priority jobs when they are queued to run. A column indicating job priority is available for display in the Saved Jobs view.
Use Online Archive	<p>Set read and write access to a source table while the table is copied to the target:</p> <ul style="list-style-type: none"> • Default: Use the daemon setting in the Data Mover Setup portlet. • True: Applies only to Teradata DSA jobs. Update the source table during the copy process, but the changes are not transferred to the target table. Instead, the data contained in the target table matches the data that was in the source table at the beginning of the copy process. • False: Do not use the online archive. <p>Online archiving demands more processing so expect job performance to be reduced.</p>
Logging Level	<p>Set the amount of detail provided for job logging as follows:</p> <ul style="list-style-type: none"> • Off: Disable logging. • 1: Enable minimal logging. Provides essential information, such as error and warning messages. • 2: Enable more logging. Provides additional information, including informational messages for example. • 99: Enable full logging. Provides detailed information, including debug messages for example.
Allow Overwrite	<p>Set to True overwrite tables and objects on the target automatically, if the target table or object exists.</p> <p>The following describes the behavior when moving join indexes, hash indexes, and triggers to Teradata Database when set to True:</p> <p>Same Name Join or Hash Index If the join or hash index of the source database is named the same as that in the target database, it is overwritten.</p> <p>Table with associated Join or Hash Index Moving a table that exists on the target with an associated join or hash index overwrites the target table and recreates the join or hash index on the newly copied table.</p> <p>Trigger Exists with Same Name If a trigger exists on the target database with the same name as in the source and within the same database, it is overwritten.</p> <p>Table Exists with Trigger If a table exists on the target with a trigger, the target table is overwritten. The trigger is recreated on the newly copied table.</p>
Compare DDL	<p>Set to compare the DDL of source tables and views with those of their existing target tables and views. Views are compared only if their copyData attributes have been specified as true.</p> <ul style="list-style-type: none"> • Default: Use the daemon setting in the Data Mover Setup portlet. • True: Compare the DDL of source tables and views to those of their corresponding target tables and views.

Field	Description
	<ul style="list-style-type: none"> • False: Do not compare the DDL of source tables and views to those of their corresponding target tables and views.
Database to Client Encryption	<p>Set to enable encryption for a particular job, table, or view using any Data Mover interface.</p> <ul style="list-style-type: none"> • Default: Use the daemon setting in the Data Mover Setup portlet. • True: Copy statistics for all tables, indexes (hash or join), and underlying objects at the job level • False: Use object-level settings for copying statistics.
Copy Stats	<p>Set to enable copy the statistics for a particular job or table using any Data Mover interface.</p> <ul style="list-style-type: none"> • Default: Use the daemon setting in the Data Mover Setup portlet. • True: Override the daemon setting and enable encryption. • False: Override the daemon setting and disable encryption.
Event table logging	<p>Set logging of events related to the job by clicking the Edit button and selecting one or more event tables.</p> <p>The options are displayed if event tables are specified in the Data Mover Setup portlet.</p>
Job Permissions	Set the read, write, and execute permission for users and roles.
Freeze job steps	<p>Set to verify any changes in the source and target environments between the time the job was first created and the time the job is run. If you select this option you can later update job steps from the Saved Jobs view.</p> <p>Note: This option generally improves performance.</p>
Foreign Server Name	Select the foreign server that was created to copy data between Teradata and Teradata systems.
Cloud Staging Area	Select the cloud staging area that was created to use as staging in the cloud to copy data between Teradata systems.
Query Band	Set to tag sessions or transactions with user-defined name-value pairs in the format of <i>key=value</i> as comma-separated name pairs. If you define a query band for a job, Data Mover uses that definition for all transactions on the source and target systems for the job.
Target Database	Set a default target database that is used for all database objects included in the job request. Click ▾ to select a database. You can use the filter to display only the items that match the characters you enter.
Source staging Database for tables	Set a default source staging database that is used for all tables included in the job request. Click ▾ to select a database. You can use the filter to display only the items that match the characters you enter.
Target staging Database for tables	Set a default staging database for tables that is used for all tables included in the job request. Click ▾ to select a database. You can use the filter to display only the items that match the characters you enter.

Field	Description
Target staging Database for temporary artifacts	Set a default staging database for temporary artifacts, such as work tables, log tables, error tables, or macros. Click ▼ to select a database. You can use the filter to display only the items that match the characters you enter.

About Advanced Job Settings

You can select advanced job save options from the **Job Settings** tab. Click **Advanced** to access job performance settings. Data Mover provides default values for these settings.

Teradata Systems

For Teradata systems, you can select these advanced job save options:

Data Streams

- For DSA jobs, specify the number of streams per database node. If no value is specified, Data Mover does not generate a default value and DSC determines the soft stream limit at run time.
- For Teradata PT API jobs, specify the number of data streams to use between the source and target databases. If no value is specified, Data Mover generates a default value based on the size of the tables being copied and the number of available AMPs.
- All other utilities use a single data stream.

Source Sessions

Specifies the number of sessions per data stream for the source system. The following occurs when no value is specified:

- For TPT jobs, a default value is generated based on the size of the tables being copied and the number of AMPs available.
- Settings do not apply to jobs using other copy methods such as DSA.

Target Sessions

Specifies the number of sessions per data stream for the target system. The following occurs when no value is specified:

- For TPT jobs, a default value is generated based on the size of the tables being copied and the number of AMPs available.
- Settings do not apply to jobs using other copy methods such as DSA.

Max Agents per Task

Specifies the maximum number of agents that Data Mover allocates at the same time to one task in jobs that use the Teradata PT API. If multiple agents are installed in the Data Mover environment, you can enter an integer value greater than one to improve performance for a job that copies large amounts of data. If you do not provide a value for **Max Agents per Task**, Data Mover dynamically calculates a value at runtime.

Force Utility

Forces Data Mover to use a specific Teradata utility or API operator for the copy job. Data Mover automatically selects the best utility for the job.

Source Character Set

Specifies the session character set that is used to communicate with the source system.

Target Character Set

Specifies the session character set that is used to communicate with the target system.

Target Group Name

Specifies a shared pipe target group to run DSA jobs instead of having Data Mover automatically select one. If the specified target group does not exist, the job fails.

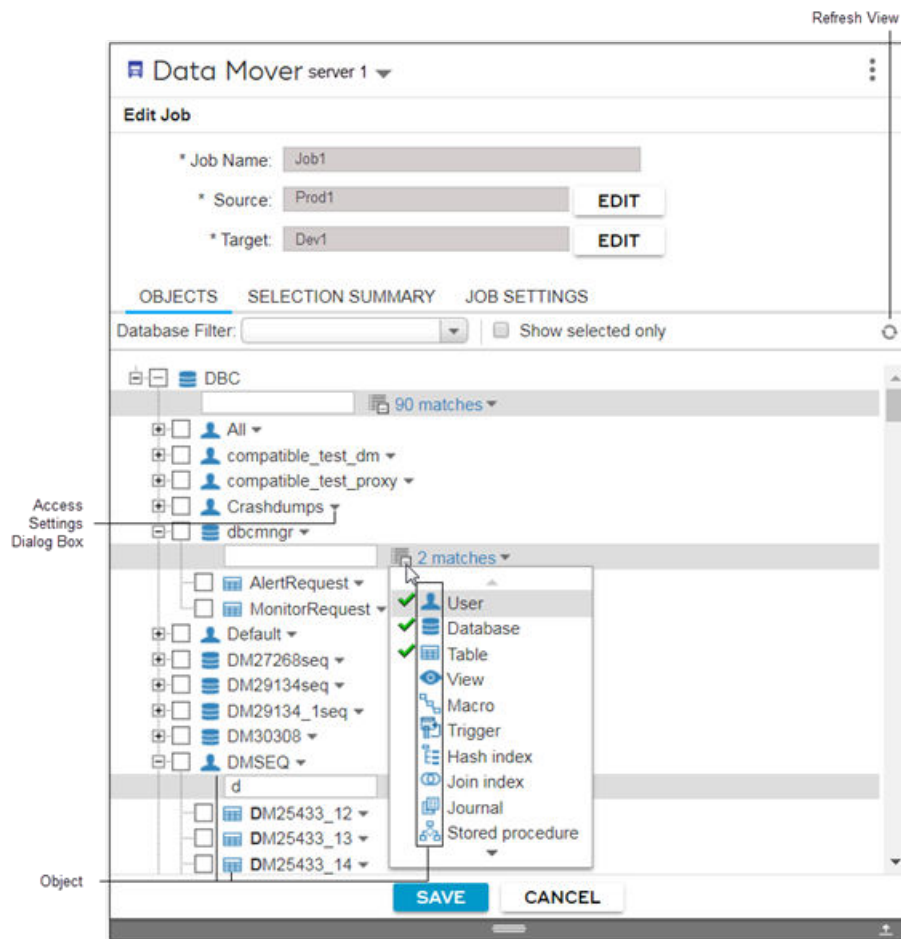
Parallel Builds

Specifies the number of tables with indexes that can be built concurrently when using DSA. The maximum number of concurrent builds is 5 (default value).

Object Browser

The object browser lets you view a list of objects on a source Teradata Database system and select objects to copy to the target system.

The object browser displays database objects in the connected system as a hierarchically organized tree. Displayed database objects include databases, users, foreign servers, tables, views, macros, triggers, stored procedures, join indexes, and hash indexes. You can use the top level **Database Filter** to filter on databases and users, and filters at the branch levels to limit the number of objects displayed in the tree.




Control	Action
Database Filter	Enables you to enter text and select from a list of databases and users that contain the text you entered to navigate directly to them. This is as an alternative to drilling down through of the branches of the tree to locate the database or user.
Show Selected Only check box	Toggles the view so that only the objects you selected are displayed. You must expand DBC, the first level of the object browser tree, to see the selected objects.
Settings For Data Mover, General tab	Provides options for selecting the default object types to display in the object browser
Refresh view	Refreshes the object browser view.
Object icons	Identifies the database object.
Database Settings dialog box	Provides options for copying database objects.
User Settings dialog box	Provides options for copying user objects.
View Settings dialog box	Provides options for copying views.

Control	Action
Table Settings dialog box	Provides options for copying tables.
Hash Index Settings dialog box	Provides options for copying hash indexes.
Join Index Settings dialog box	Provides options for copying join indexes.
Trigger Settings dialog box	Provides options for copying triggers.
Column Settings dialog box	Provides options for changing the target name and destination type on a column.

Setting Default Objects










The object browser displays database objects based on the default settings of the user's role. You can configure the object browser to display different object types. The tree refreshes quicker if you select fewer object types.





1. Click  in the portlet frame and select **Settings**.
2. In the **General** tab, select the object types to display by default in the object browser.
3. Select **OK**.

Object Types

When selecting object types for copying, you can select any *independent* object by itself or with other objects. You can select a *dependent* object only if another object such as a table or database is also selected.

The following database objects and associated icons are displayed in the object browser tree.

Database Object	Icon	Independent or Dependent Object
User		Independent
Database		Independent
Schema		Independent on associated Teradata Database
Macro		Independent
View		Independent
Stored Procedure		Independent
Table		Independent
Column		Dependent on associated table
Trigger		Dependent on associated table

Database Object	Icon	Independent or Dependent Object
Hash Index		Independent
Join Index		Independent
Function Aliases		Independent
Foreign Server		Independent

Supported Objects During Moves Between Databases

The following database objects are available in the object browser tree:

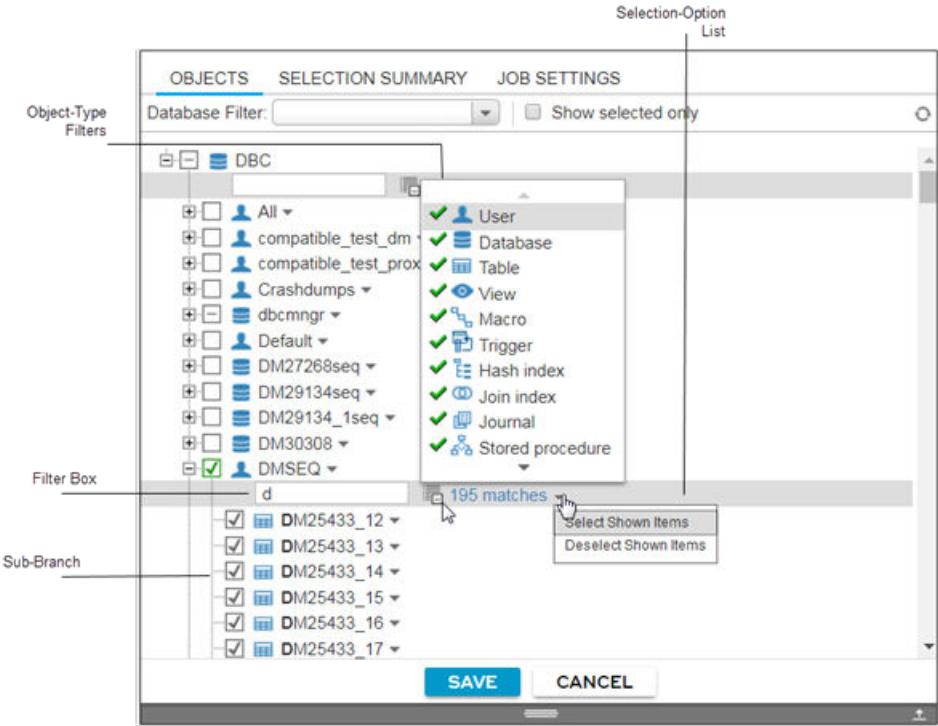
When selecting object types for copying, you can select a subset of the viewable objects.

Components Available	Teradata to Teradata
View Objects	<ul style="list-style-type: none"> • Database • Users • Tables • Most Object Types ¹
Select Objects	<ul style="list-style-type: none"> • Database • Users • Tables • Most Object Types ¹
Relocate to Target	Database
Table Settings	Yes
Column Settings	No
Foreign Server Objects	Yes
Function Alias Objects	Yes
¹ Most Teradata Database object types are viewable and selectable.	

Using the Object Browser

When creating a new job or editing an existing one, use the object browser to locate and select the objects you want to copy. Filtering allows you to display only database objects that match your filter criteria. The filtering controls are identified in the following graphic and explained in the table.

1. Use the top level **Database Filter** or expand the branches and select the object types for your job.



Control	Action
Database Filter	Filters the entire tree to display the databases and users that match the text you enter in the box. Filtering is case insensitive.
Filter Box	Filters for database objects in the branch or sub-branch that match the text you enter in the box. Filtering is case insensitive.
Check Box	Selects all objects in the branch and sub-branch.
Object-Type Filter	Selects types of database objects.
Matched Objects	Total objects that match the current filter criteria in the branch. If no objects of the type are found, the branch displays 0 matches.
Selection-Option List	Toggles selection of multiple items in the sub-branch only.
Page-Results Navigator	Navigates through pages of filtered results in the branch.

2. Click **Save**.
The **Saved Job** view appears.

Data Mover RESTful API

Data Mover RESTful API

Data Mover RESTful APIs provide an interface to Data Mover functions using standard HTTPS requests. You can access the Data Mover RESTful service using any standard REST client. For example, you can use REST client modules provided in scripting languages like Python or PERL, or those provided in visual tools like the Firefox RESTClient add-on.

Data Mover RESTful API returns status links in response to certain requests. When using RESTful API through a proxy service instead of Data Mover directly, you can modify these links to reference the proxy service host rather than the Data Mover host. This is achieved by adding `hostname:port` to the Data Mover **REST Accept Host List** and adding the `X-Forwarded-Host: hostname:port` header to the request.

If your proxy service uses SSL and your Data Mover provider does not, add the `X-Forwarded-SSL: on` header to prefix the Data Mover links with HTTP instead of HTTPS.

Data Mover RESTful API Calls

Calls to the Data Mover RESTful API service involve the following elements:

- HTTPS is enabled by default.
- All calls to the Data Mover RESTful interface return an HTTPS status code that indicates whether the call was successful or not.

For more information, see [RESTful API Status Codes](#)

- Some calls to the Data Mover RESTful Interface return information in the body of the response in JSON format.
- A URL identifying the Data Mover server on which the REST service is running such as, `https://dmprod:1443/datamover/jobs.dmprod` where each part of the path defines the following:
 - `dmprod` – the DNS alias of the Data Mover server
 - `1443` – the default port on which the REST service is listening
 - `/datamover/jobs` – the REST resource being referenced
- Each call to the Data Mover RESTful API is made using a specific HTTPS method: GET, POST, PUT, or DELETE. Using different HTTPS methods on the same REST resource performs different actions, such as the following examples:
 - Using GET on `/datamover/jobs` returns a list of all Data Mover jobs
 - Using POST on `/datamover/jobs` creates a new job

All calls to the Data Mover RESTful interface must include an Accept Header with the version of the API being used. The header provided is a key-value pair. The key is Accept and the current version value is `application/vnd.com.teradata.datamover-v1.0+json`.

- Some calls to a Data Mover RESTful API require an input parameter. Input parameters are either specified in the body of the request in JSON format or provided in the URL path itself.

An example of the URL path is, `https://dmprod:1443/datamover/jobs/job-name?outputLevel=4` where **outputLevel=4** is a parameter.

Data Mover REST Accept Host List

All requests sent to the Data Mover REST API must use a host defined in the Data Mover REST Accept Host List. This applies to the Host header and X-Forwarded-Host header. If the values for either header does not match one of the hosts defined in the Accept Host List, the request is rejected. By default, the list contains `localhost:1443`. Leave this value in the list unless you do not want to accept calls to Data Mover REST using `localhost:1443`.

In addition to the values in the Accept Host List, Data Mover dynamically looks up the server name and server address of the host where Data Mover REST is running and accepts these values. These values are accepted as *server name*:1443 and *server address*:1443. To view the dynamic server name and server address being used, see `dmRest.log`.

Adding Additional Hosts to the Accept Host List

- From the `tdmrest.properties` file, add the host name to the `accept.host.list` property.
Host values can be provided in *host name* format or *host name:port* format. If multiple hosts exist, separate each *host name* with a comma.
`accept.host.list=localhost:1443`
`accept.host.list=localhost:1443, host1:5555, host2, host3`

Note:

Use the format that exactly matches the format used in your requests. If they do not match, requests are not accepted.

- Restart the `tdmrest` service:
`/etc/init.d/tdmrest stop`
`/etc/init.d/tdmrest start`

Supported Data Mover Commands for RESTful API

Actions	Requires Input Parameters	Parameter Location	HTTP Method
Backup Daemon	No	Request Body	POST

Actions	Requires Input Parameters	Parameter Location	HTTP Method
Check Permissions	No	Request Body	POST
Cleanup Job	No	–	DELETE
Create Event Table	Yes	Request Body	POST
Create Job	Yes	Request Body	POST
Create Policies	No	Request Body	POST
List Data Mover Information	No	–	GET
Delete Event Table	No	–	DELETE
Delete Job	No	Path	DELETE
Edit Event Table	No	Request Body	PUT
Edit Job	Yes	Request Body	PUT
Encrypt Password	Yes	Request Body	POST
List Agents	No	–	GET
List Configuration	No	–	GET
List Event Tables	No	–	GET
List Job Definition	No	–	GET
List Jobs	Yes	–	GET
List Job Steps	No	Path	GET
List Policies	Yes	Path	GET
List Tasks	Yes	Path	GET
Move Job	Yes	Request Body	POST
Restart Job	Yes	Request Body	POST
Restore Daemon	No	Request Body	POST
Save Configuration	Yes	Request Body	PUT
Start Job	Yes	Path and Request Body	POST
Status	No	Path	GET
Stop Job	No	–	DELETE
Update Job Priority	Yes	Request Body	PUT
Update Job Steps	No	Request body	PUT
¹ List jobs variants are startTimeAfter, endTimeBefore, and endTimeAfter.			

Data Mover RESTful Service

During installation of Data Mover, the DM REST component is installed and started automatically. If you need to stop or restart services, use the following commands:

Command	Description	Example
start	Starts the service. If the service is already running, a new instance is not started.	To start the Data Mover REST service, type: <code>/etc/init.d/tdmrest start</code>
stop	Stops the service.	To stop the Data Mover REST service, type: <code>/etc/init.d/tdmrest stop</code>

Viewing Data Mover REST Samples

The following table lists the REST samples added as part of the Data Mover REST package. If an input file is required, the name is provided.

Data Mover Command	REST Sample Script File	Input File Used by Script
cleanup	cleanup.py	–
create	createjob_td2td.py	createjob_td2td.json
delete_job	delete_job.py	–
edit	edit_job.py	createjob.json
list_agents	list_agents.py	–
list_job_definition	listjobdefinition.py	–
list_jobs	listjobs.py	–
list_job_steps	list_job_steps.py	–
list_tasks	list_tasks.py	–
restart	restart_job.py	–
save_configuration	saveconfiguration.py	saveconfiguration.json
start	startjob_td2td.py	startjob_td2td.json
status	jobstatus.py	–
stop	stop_job.py	–
update_job_steps	update_job_steps.py	–

1. On the Data Mover server, locate the `/opt/teradata/datamover/rest/tdmrest/samples` directory.
2. Run the `sh get_cert` command to get the public key of the self-signed Data Mover host.

3. Run the `python runsamples.py` command and enter the script name you want, such as `createjob_td2td.py`, when prompted.

For additional information, refer to the README file found in the `/opt/teradata/datamover/rest/tdmrest/samples` directory.

Swagger User Interface

Data Mover supports the Swagger documentation and user interface. The Swagger interface provides details about RESTful API specification and you can test REST APIs using Swagger. Once enabled, the Swagger interface can be constructed as in the following example:

```
https://dm_host:1443/datamover/swagger-ui.html
```

Use the following commands to enable or disable Swagger:

Option	Description
Enable	Set <code>swagger.ui.enabled=true</code> in <code>tdmrest.properties</code> .
Disable	Set <code>swagger.ui.enabled=false</code> in <code>tdmrest.properties</code> .

Note:

You must restart the REST service after enabling or disabling Swagger.

Monitoring Teradata Data Mover

Using Teradata Ecosystem Manager to Monitor Jobs

Teradata Ecosystem Manager can be used to monitor Data Mover jobs. It is not enabled by default. To enable this feature, perform the following:

1. Run the `list_configuration` command.
2. In the resulting XML file, modify the `tmsm.mode` property. Change value to BOTH, ONLY_REAL_TMSM, or ONLY_INTERNAL_TMSM.
3. Run `save_configuration` to save the configuration.
If `tmsm.mode` is set to ONLY_REAL_TMSM, Teradata Ecosystem Manager is notified of all events. If `tmsm.mode` is set to ONLY_INTERNAL_TMSM, events are stored in the TMSMEVENT table.

The following is an overview of the events to expect as a job runs:

- As a job starts, START events with ResourceType DM_PROCESS_SOURCE and DM_PROCESS_TARGET are logged.
- A job can contain multiple tasks. START events are logged for each task, with different ResourceTypes corresponding to the type of task being performed
- Some long-running task types provide additional information with STEP events. The AffectedDatabaseName and AffectedTableName columns indicate the object being worked on. Incremental status information is stored in the UOWHealthStr column. UOWHealthAmt remains 0 until actions on an object complete, at which time UOWHealthAmt indicates the number of rows copied. The UOWHealthAmt should match on the source and target. A mismatch usually indicates a problem.

Note:

Tasks with type of SELECT_INSERT report only the number of bytes copied for large objects (LOBs) in the UOWHealthStr column. It is normal to see 0 bytes copied for tables that do not contain any LOBs for SELECT_INSERT tasks.

- At the end of a task, END events are logged with the total number of rows and bytes copied.
- At the end of a job, END events with ResourceType of DM_PROCESS_SOURCE and DM_PROCESS_TARGET are logged, and a HCHK event indicates successful completion.
- If the job cannot complete successfully, an ALERT event is logged with the error message in the EventMsg column and corresponding error code in the AlertCode column.

Teradata PT API sends its own events to Teradata Ecosystem Manager by default. To prevent sending duplicate events to Teradata Ecosystem Manager, Data Mover prevents Teradata PT API from sending its own events when the utility runs in Data Mover jobs. In those situations, only the Data Mover daemon is able to send events to Teradata Ecosystem Manager.

The events that are sent to Teradata Ecosystem Manager can also be sent to an external SQL table for additional processing. For more information, see [Using Event Tables](#).

About TMSMEVENT Table Columns

The following table describes columns of the event table (TMSM table):

Column Name	Description	Sample Values																						
ResourceId	Base name of the Data Mover job running, without a timestamp.	sample_job_1																						
ResourceType	<div><p>A string that describes the task type, the event, and the location. Use <i>HBEAT</i> for the health check.</p><p>The following table describes the base names of different task types:</p><table><tr><th>Task Type</th><th>Base Name</th></tr><tr><td>DSA</td><td>ARCHIVE_COPY</td></tr><tr><td>SQL</td><td><ul style="list-style-type: none">CLEAN_UPPRE_DATA_MOVEINSERT_SELECTCOPY_STATSVALIDATINGSQL_CREATE</td></tr><tr><td>TPT</td><td><ul style="list-style-type: none">EXPORT_LOADEXPORT_UPDATEEXPORT_STREAMTPTAPI</td></tr><tr><td>JDBC</td><td>SELECT_INSERT</td></tr><tr><td>T2T</td><td>T2T</td></tr><tr><td>Compare DDL</td><td>COMPARE_DDL</td></tr><tr><td>Object Verification</td><td>OBJECT_VERIFICATION</td></tr><tr><td>Row Count Validation</td><td>ROW_COUNT_VALIDATION</td></tr><tr><td>Missing Object Alert</td><td>OBJECT_SKIPPED</td></tr><tr><td>Health Check</td><td>HBEAT</td></tr></table></div>	Task Type	Base Name	DSA	ARCHIVE_COPY	SQL	<ul style="list-style-type: none">CLEAN_UPPRE_DATA_MOVEINSERT_SELECTCOPY_STATSVALIDATINGSQL_CREATE	TPT	<ul style="list-style-type: none">EXPORT_LOADEXPORT_UPDATEEXPORT_STREAMTPTAPI	JDBC	SELECT_INSERT	T2T	T2T	Compare DDL	COMPARE_DDL	Object Verification	OBJECT_VERIFICATION	Row Count Validation	ROW_COUNT_VALIDATION	Missing Object Alert	OBJECT_SKIPPED	Health Check	HBEAT	<ul style="list-style-type: none">SELECT_INSERT_TARGETHBEATCOPY_STATS_TARGETROW_COUNT_VALIDATION_TARGETDM_PROCESS_SOURCEROW_COUNT_VALIDATION_SOURCESQL_CREATE_TARGETDM_PROCESS_TARGETSELECT_INSERT_SOURCEARCHIVE_COPY_SOURCEARCHIVE_COPY_TARGETPRE_DATA_MOVE_TARGETINSERT_SELECT_TARGET
Task Type	Base Name																							
DSA	ARCHIVE_COPY																							
SQL	<ul style="list-style-type: none">CLEAN_UPPRE_DATA_MOVEINSERT_SELECTCOPY_STATSVALIDATINGSQL_CREATE																							
TPT	<ul style="list-style-type: none">EXPORT_LOADEXPORT_UPDATEEXPORT_STREAMTPTAPI																							
JDBC	SELECT_INSERT																							
T2T	T2T																							
Compare DDL	COMPARE_DDL																							
Object Verification	OBJECT_VERIFICATION																							
Row Count Validation	ROW_COUNT_VALIDATION																							
Missing Object Alert	OBJECT_SKIPPED																							
Health Check	HBEAT																							

Column Name	Description	Sample Values
TDPID	Host name for source or target system depending on where the job is running.	<ul style="list-style-type: none"> source_12838 target_83838
UOWId	Alternate ID or name for the batch of work associated with the job. If you provide a value for this parameter, Data Mover reports this value as the unit of work ID when sending events to Teradata Ecosystem Manager or to its internal TMSMEVENT table. If you do not specify this parameter, Data Mover uses a default value as the unit of work ID when sending events to Teradata Ecosystem Manager or to its internal TMSMEVENT table. The default value for the unit of work ID is composed of the job execution name and the current timestamp. For example, if you want to define the origins of a query source the job execution name is sales_table, the default value of the unit of work ID is sales_table-20211110122330.	sample_job_1-20211110122330
EventType	Row event type For example, <i>HCHK</i> stands for Health check.	<ul style="list-style-type: none"> START STEP END ALERT HCHK
EventTS	<i>DateTime</i> string representing the event logging time in the Event table.	2021-11-10 12:24:04.689000
EventDate	<i>Date</i> string representing the event logging time in the Event table.	2021-11-10
CreatedTS	<i>DateTime</i> string representing the event logging time in the Event table.	2021-11-10 12:24:04.689000
EventMsg	Event description depending on the type of the event happening. 200 characters is the limit for this column.	<pre>[TaskID:308] ERROR: Failed SQL: ----- CREATE MULTISET TABLE database1622651320374. Table16226513204_ee3988_ t ,NO FALLBACK , NO BEFORE JOURNAL, NO A Replication Completed.</pre>

Column Name	Description	Sample Values				
		<div>ROW_COUNT_VALIDATION is starting.</div> <div>SQL has completed creating the object database1622651320374.sp1622651320438</div>				
UOWHealthAmt	Number of processed rows For example, when archiving a table, this column reports the number of rows exported. The value can also be null.	100				
UOWHealthStr	A string that describes the number of bytes and rows processed as part of the task.	2 rows 84 bytes				
AlertCode	Associated error code For example, 3870 indicates the table does not exist, and Null indicates rows have no errors.	13525				
SeverityLvl	Value of severity level Set the column value to 10 for the alerts. For other events, set the column value to 1 or keep it empty.	<ul style="list-style-type: none">• 10• 1• Keep empty.				
EventSourceSystem	Hostname of the system Depending on the location of the event, the column value can be the hostname of the Agent or the Daemon process.	<ul style="list-style-type: none">• source_12838• target_83838				
EventSourceUser	OS username used by Data Mover Depending on the location of the event, the column value can be the username used for the Agent or the Daemon process.	dmuser				
JobName	Job execution name	sample_job_1-20211110135117				
JobStep	ID of the current event step.	1, 2, 3,...				
EventDML	Similar to ResourceType column describing specifically the type of event happening without the trailing string <code>_source</code> and <code>_target</code> .	<div>Sample ResourceType to EventDML mappings:</div> <table><tr><th>ResourceType</th><th>EventDML</th></tr><tr><td>HBEAT</td><td>null</td></tr></table>	ResourceType	EventDML	HBEAT	null
ResourceType	EventDML					
HBEAT	null					

Column Name	Description	Sample Values	
		ResourceType	EventDML
		SELECT_INSERT_SOURCE	EXPORT_BYTES
		SELECT_INSERT_TARGET	INSERT_BYTES
		DM_PROCESS_SOURCE	DM_PROCESS
		COPY_STATS_TARGET	COPY_STATS
		SELECT_INSERT_SOURCE	EXPORT
		SELECT_INSERT_TARGET	INSERT
		ROW_COUNT_VALIDATION_SOURCE	null
		SQL_CREATE_TARGET	CREATE
		ARCHIVE_COPY_SOURCE	ARCHIVE
		ARCHIVE_COPY_TARGET	COPY
Optional1	Not used - Set to <i>Null</i>	Null	
Optional2	Not used - Set to <i>Null</i>	Null	
Optional3	Set to 1 for non health check steps, and <i>Null</i> for health checks.	1 and Null	
StateCode	Not used - Set to <i>Null</i>	Null	
ApplicationId	Set to <i>DataMover</i> string. Indicates that the row is inserted by Data Mover.	DataMover	
AffectedDatabaseName	Parenting database name of the processing object.	database1622651320297	
AffectedTableName	Name of the object that is processed as part of the step.	<ul style="list-style-type: none"> Table1622651320350 sp1622651320362 macro1622651320358 	

Column Name	Description	Sample Values
		• View1622651320430
LogFile	Not used - Set to <i>Null</i>	Null
UOWClass	Job execution name	sample_job_1-20211110122330
UOWName	Not used - Set to <i>Null</i>	Null
UOWDesc	Not used - Set to <i>Null</i>	Null
UOWTS	Not used - Set to <i>Null</i>	Null
UOWFromTS	Not used - Set to <i>Null</i>	Null
UOWDate	Not used - Set to <i>Null</i>	Null
UOWSourceSystem	Hostname of the system where the daemon is running.	mysystem12232
ModifiedTS	<i>DateTime</i> string representing the event logging time in the Event table.	2021-11-10 13:52:04.050000
MDA_VERSION	Metadata version which is set to 101.	101

Using Data Mover to Monitor Status

Data Mover provides status monitoring through the command-line interface `status` command. For more information, see the [status](#) command section.

Each Data Mover component maintains rolling log files for internal health monitoring. They are `dmAgent.log`, `dmCommandLine.log`, and `dmDaemon.log` for the agent, command line, and daemon, respectively. The maximum size for these files is 10MB. When 10MB is reached, up to three additional backup files are created. For example, `dmAgent.1`, `dmAgent.2`, and `dmAgent.3` are created for the agent. Similar backup files are created for the command line and daemon.

Do not delete the log files. You use them to diagnose problems, should they arise.

Performance Monitoring and Tuning

Monitoring Job Performance

Elapsed time, row count, and byte count is reported for each Data Mover job that is run. If Teradata Ecosystem Manager is not available, view this information using the event table. Row count is available only at the end of the job, but byte count is reported periodically as the job runs. The frequency in which the byte count is reported is controlled with the `tmsm.frequency.bytes` configuration parameter, which can be modified using the command-line interface's `list_configuration` and `save_configuration` commands.

- When Teradata JDBC is used to copy table data, only the byte count of large object (LOB) column data is reported. If the table does not contain any LOB columns, no byte count is reported. If a table contains LOB columns and non-LOB columns, the byte count reported for that table reflects only the byte count of the LOB column data copied.
- Setting the `tmsm.frequency.bytes` parameter to a small number can lead to performance degradation due to the increase in status reports.

Data Mover Best Practices

A comprehensive implementation and configuration best practices guide is available for Data Mover. This guide includes networking information to help you understand and resolve a variety of performance issues, including test and validation procedures for suggested changes.

The Data Mover Best Practices Guide is available online and for download at [Teradata® Data Mover Best Practices Guide](#).

About Tuning Data Mover

Data Mover can be tuned by setting property values that:

- Limit the number of concurrent jobs that can run on the daemon at the same time
- Increase the number of Data Mover agents
- Limit the maximum load slots used on the target system
- Increase the number of sessions
- Use the merge table feature

Limiting the Number of Concurrent Jobs

Each Data Mover job that is currently running has an effect on the ability to run other jobs. A Data Mover job has one or more tasks that an agent can perform using Teradata JDBC, DSA, QueryGrid, or Teradata PT API. As the number of concurrent jobs increases, the number of tasks increases, resulting in competition

for resources of the agent. To avoid a bottleneck, Data Mover limits the number of jobs that can run concurrently. After the limit is reached, all new job execution requests are queued and run when other jobs complete.

By default, the concurrent task limit is set to 20. Depending on the types of jobs being run and the number of Data Mover agents available, this limit could be too low or too high. You may want to change it. You can also use flexible scheduling to improve performance by setting different limits for different time periods during the day.

1. Set the value for the `jobExecutionCoordinator.maxConcurrentJobs` parameter in the `daemon.properties` file to the desired limit.
This property is updated dynamically. The settings you provide take effect one minute after you save `daemon.properties`. The daemon does not need to be restarted.

About Increasing the Number of Agents

Each agent is capable of processing five tasks at one time. Increasing the number of agents allows more tasks to be processed at one time, increasing the number of Data Mover jobs processed at one time. In some cases, this improves individual Data Mover job performance.

Data Mover jobs that use Teradata PT API or Teradata JDBC have separate tasks for each table being copied. Increasing the number of agents improves the job performance by allowing more of the job tables to be copied concurrently.

You may also want to limit the maximum number of current tasks that an agent can run by setting the `agent.maxConcurrentTasks` parameter in the `agent.properties` file. This parameter is set dynamically and does not require that the agent service be restarted. If you update the value, it takes effect one minute after the updated file is saved. In addition, you can also use flexible scheduling to set different concurrent limits depending on the time of the day.

About Flexible Scheduling

You may want to limit the number of concurrent and queued jobs to values that vary depending on the time of day. Flexible scheduling loads different property values based on the system timestamp for the same property. The following properties support flexible scheduling:

- **`jobExecutionCoordinator.maxConcurrentJobs`**
- **`jobExecutionCoordinator.maxQueuedJobs`**
- **`agent.maxConcurrentTasks`**

You can specify hourly ranges for values by using the format for flexible scheduling: `<property name>.<hour> = value`. Specify a number for `<hour>` using a 24 hour format. For example, to specify 3 p.m., type 15.

When flexible scheduling is used in the `daemon.properties` file, the default value for `jobExecutionCoordinator.maxConcurrentJobs` or `jobExecutionCoordinator.maxQueuedJobs` specifies the value used starting at 12 a.m. every day. When an hourly value is specified (for example,

`jobExecutionCoordinator.maxConcurrentJobs.2`), that value is effective until the next hourly value specified or until the next day starts, at 12 a.m.

For example, suppose you wanted to schedule the maximum number of concurrent tasks to be 20 from 12 a.m. to 2 a.m., 25 from 2 a.m. to 5 a.m., and 5 from 5 a.m. to 12 a.m. The value would then go back to 20 when the next day starts at 12 a.m. You would set the values as follows:

```
jobExecutionCoordinator.maxConcurrentJobs =20
jobExecutionCoordinator.maxConcurrentJobs.2=25
jobExecutionCoordinator.maxConcurrentJobs.5=5
```

Flexible scheduling works similarly for the `agent.maxConcurrentTasks` parameter in the `agent.properties` file. In the example here, the agent can run up to 10 concurrent jobs from 12 a.m. to 5 a.m., and 5 concurrent jobs from 5 a.m. until 12 a.m. the next day.

```
agent.maxConcurrentTasks=10
agent.maxConcurrentTasks.5=5
```

About Limiting the Load Slots Used on the Target System

In the target system load slot limit feature, Data Mover limits the number of job tasks running at the same time based on their use of the Teradata PT API LOAD and UPDATE operators. These API operators, which load data to the target system, restrict the number of active jobs running at a specific time against a specific Teradata Database.

Note:

Some limits external to Data Mover can override the Data Mover sessions:

- TASM - If Teradata Active System Management (TASM) session rules are enabled on a particular source or target system, the TASM rules may override the Data Mover sessions.
 - DBSControl - If you limit the `MaxLoadTask` and `MaxLoadAWT` fields of the DBSControl utility, this could override the Data Mover sessions.
-

You can easily exceed operator limits for a specific target Teradata Database because each Data Mover job can use several instances of these operators, and multiple Data Mover jobs can run at the same time. Data Mover jobs can subsequently fail or hang as attempts to create new instances of the operators are rejected or put on hold by the Teradata Database.

To avoid such failures, Data Mover limits the number of concurrent job tasks, based on the operator being used by the tasks. The limit is enforced independently for each target Teradata Database system. For example, if a job has 10 tasks that use the restricted operators, and the target system load slot limit is 5, Data Mover allows up to 5 of the 10 tasks from that job to run concurrently against the target Teradata Database system. If 2 jobs that each has 10 tasks using the restricted operators are run against the same target Teradata Database system at the same time, Data Mover allows only 5 of the combined 20 tasks from both jobs to run concurrently. If 2 jobs are run against different target Teradata Database systems, and

the target system load slot limit is 5 for both systems, Data Mover allows 5 tasks from each job to run at the same time. Data Mover jobs and tasks that use non-restricted operators are not affected by this feature.

The target load slot feature does not affect the number of Data Mover jobs that can run at a specific time or determine which utility Data Mover chooses when creating new jobs.

The `target.system.load.slots` configuration property, which can be modified through the `list_configuration` and `save_configuration` commands, controls the target system load slot limit. Use the `target.system.load.slots` property to set the following:

- A default load slot limit for all target Teradata Database systems
- Different limits for specific target Teradata Database systems

Note:

For the target system load slot limit to work correctly, all Data Mover jobs run against a specific target Teradata Database system must use the same name or IP address for the same target system. Otherwise, Data Mover assumes that the jobs are using different target systems and limits their tasks separately, rather than collectively.

Setting the Default Target System Load Slot Limit

Data Mover is installed with a default target system load slot limit of 5. The limit applies to all target Teradata Databases that do not have specific limits set for them.

1. Run `list_configuration` for current property values.
2. Find the `target.system.load.slots` property in the configuration XML file produced by `list_configuration` and complete the following:
 - Below the default value tags, add a set of value tags.
 - Inside the opening value tag, set the `system` attribute to the name or IP address of the target Teradata Database system (the name or IP address must match the identification being used in Data Mover jobs running against this system).
 - Specify a limit for this system between the opening and closing value tags.

The following example of the `target.system.load.slots` property shows the default value set to 5.

```
<property>
<key>target.system.load.slots</key>
<value>5</value>
<description>Purpose: Controls maximum number of load slots that Data
Mover can
use at one time on target Teradata systems. Default: 5.</description>
</property>
```

3. Run `save_configuration` to save the changes.

Setting the Target System Load Slot Limit for a Specific Target System

1. Run `list_configuration` for current property values.
2. Find the `target.system.load.slots` property in the configuration XML file produced by `list_configuration` and complete the following:
 - Below the default value tags, add a set of value tags.
 - Inside the opening value tag, set the `system` attribute to the name or IP address of the target Teradata Database system (the name or IP address must match the identification being used in Data Mover jobs running against this system).
 - Specify a limit for this system between the opening and closing value tags.

The following is an example of the `target.system.load.slots` property with the default value set to 5 and a limit of 10 specified for the target Teradata Database system, `targetSystemA`.

```
<property>
  <key>target.system.load.slots</key>
  <value>5</value>
  <value system="targetSystemA">10</value>
  <description>Purpose: Controls total number of load slots that Data Mover
can
      use at one time on target Teradata systems. Default: 5.</
description>
</property>
```

3. Run `save_configuration` to save the changes.

Removing a System-Specific Target System Load Slot Limit

Note:

The default value cannot be removed.

1. Run `list_configuration` for current property values.
2. Find the `target.system.load.slots` property in the configuration XML file.
3. Delete the set of value tags for the target system.
4. Run `save_configuration` to save the changes.

About Increasing the Number of Sessions

Data Mover provides the ability to independently specify the number of source and target sessions that a job uses when transferring data. When creating jobs through the command-line interface, the number

of sessions used can be specified with the `source_sessions` and `target_sessions` parameters when using the `create` or `move` command.

Specifying a higher number of sessions can improve individual job performance, although specifying too many sessions can lead to performance degradation. The optimal number of sessions varies depending on the environment and the utility being used. Consult *Teradata® Parallel Transporter Reference* and *Teradata JDBC Driver Reference* for details.

If a value for the source and/or target sessions is not provided, Data Mover dynamically determines the number of sessions; this determination is based on multiple factors, including number of AMPs, size of objects being moved, and so forth. This dynamic value is a good starting point, but advanced users may find more ideal settings to use for each situation.

Related Information:

[About Default Sessions and Streams](#)

About Using the Merge Table Feature

In some circumstances, Data Mover uses `MERGE` instead of `INSERT-SELECT` when copying data from a staging database to a target database. The result is better performance.

The following conditions must be met for Data Mover when using `MERGE`:

- A partial table copy must be specified.
- The table cannot be a multiset table.
- The table must contain one or more primary indexes.
- All indexes must be specified as key columns.
- For PPI tables, all partitioned columns must be specified as key columns.
- The primary index on the target table does not contain an identity column.


If these conditions are not met, Data Mover uses `DELETE` and `INSERT-SELECT` to transfer data.





Note:

For PPI tables, indexes and partitioned columns must be specified as key columns. Specifying only the indexes as key columns results in a failed job execution.

Adding a Data Mover Server to the Viewpoint Monitoring Portlet

You can add Data Mover server metrics to the **Viewpoint Monitoring** portlet, including CPU Usage, Memory Buffers, Memory Free, and many more. The `tmsmonitor` service must be running on the Data Mover server to view the metrics.

1. In the **Viewpoint Administration** view, select **Monitored Systems**.
2. Click  next to **Systems**, and select **Add Managed System**.

3. Enter the system and login information.
4. Click **Apply**.
5. In the **Teradata Viewpoint** portal page , click  next to **Add Content**.
6. Click **Viewpoint Monitoring**, and then click **Add**.
The **Viewpoint Monitoring** window appears on the page, with the **CPU System: percent** metric included by default.
7. Click  in the portlet frame and select **Settings**.
8. Click  to add a metric to the **Viewpoint Monitoring** portlet.
9. [Optional] Click  to remove a metric from the **Viewpoint Monitoring** portlet.
10. Select **OK**.

High Availability

Automatic Failover

Automatic failover is supported when two Data Mover clusters are configured in an active-standby configuration. To enable automatic failover, two additional monitoring servers are required to monitor the active and standby components. The services perform the following functions:

- The monitoring service uses SSH connections to monitor the designated-active (or primary) components to see if services are running, including the sync monitor.
- The sync monitor makes sure that the Data Mover repositories are in synchronization by monitoring Postgres replication between the active and standby servers.

A failover sequence is initiated when any of the following designated-active components are unavailable:

- Daemon
- Repository
- Agents
- REST service
- ActiveMQ

As part of the failover process, the following occurs:

- Active components stop
- Standby components start
- Monitoring service for the designated-active components stop
- Monitoring service on designated-standby components start

Note:

See the *Teradata® Data Mover Installation, Configuration, and Upgrade Guide for Customers*, B035-4102 for information on the following DSA restrictions and related configurations:

- Enabling failover for jobs using the DSA utility, pre-configuring BAR NCs for the standby DSC environment (applicable regardless of database version).
 - Pre-configuring source and target databases using Teradata systems version 16.00 or later for the standby DSC environment.
 - Configuring source and target Teradata databases with version 16.00 or earlier when the active and standby Data Mover daemons are using the bundled DSC setup. When using multiple DSC environments to run jobs, you must reconfigure DSMAIN each time you want to run the same job after a Data Mover failover occurs.
-

Failover Process

Use the `dmcluster` service script, found in `/opt/teradata/client/nn.nn/datamover/failover/`, to run commands on the failover cluster. When `dmcluster config` completes the configuration of the cluster, the following occurs:

- Services are stopped and restarted in active mode on the designated-active servers
- Standby components are stopped
- Active component servers are monitored at regular intervals by the monitoring service through an SSH connection making sure services are running

If the active Data Mover daemon, REST service, repository, ActiveMQ, or all agents used by the active daemon are unavailable, the following occurs:

- A failover sequence is initiated
- The failover sequence stops the Data Mover daemon, REST, agents, ActiveMQ, and bundled DSA (if applicable) on the designated-active servers and starts these components on the designated-standby server

When the failover sequence completes, the monitoring service shifts to the standby monitoring server and begins monitoring the now-active standby components. If the monitoring service finds a running Data Mover component or process on the standby server, it automatically stops to prevent two services from being active at the same time.

When the Data Mover cluster is actively running on the designated-standby servers, the high availability service is limited with the following changes:

- The Data Mover repositories are no longer kept in synchronization, the synchronization service is disabled and the sync monitoring service is not in use.
- No additional failover occurs when the main components are unavailable. The cluster must be manually switched back to bring the cluster [back to the original active state](#).

The `dmFailover.log` file on the monitor servers provide more information on the failover sequence.

Configuring Data Mover to Log to Server Management

Configure Data Mover to log to the Server Management by using the `tvi.useLogger` property in the `agent.properties` and `daemon.properties` files. The property defaults to `True`, making Server Management logging automatic and enabling critical failures to be immediately reported to Teradata.

TVI alerts are only sent when CMIC is configured.

Note:

Data Mover components load CMIC configuration properties in `sm_config.txt` on startup only. Restart Data Mover components to pick up changes in `sm_config.txt` after configuration.

1. Log on to the agent and daemon servers.

- Set the `tvi.useLogger` property in the `daemon.properties` file and the `tvi.useLogger` property in the `agent.properties` file to `True`.

Enabling Alerts When Failover Occurs

Server Management or TVI alerts are delivered in the form of a JSON report when failover occurs due to unavailable components.

- Log on to the active and standby monitoring servers: `local.monitor.host` and `remote.monitor.host`.
- On `/opt/teradata/client/nn.nn/datamover/failover/monitor.properties`, do the following:
 - Set `tvi.useLogger` to `True`.
 - Confirm `tvilogger.properties` exists and is configured with the correct server management logging method.

Verify Cluster Status

Use the `dmcluster status` command to check the status of a failover cluster.

Active Mode

When Data Mover services run on the designated-active servers and are monitored by the designated-active monitor, the cluster is in *active mode*. When checking the status of an active cluster with the `dmcluster status` command, the following displays:

LOCAL CLUSTER		
COMPONENT	HOST NAME	STATUS
DM Daemon	DM1	RUNNING
ActiveMQ	DM1	RUNNING
DM Monitoring Service	VP1	RUNNING
DM Sync Monitor	DM1	RUNNING
DM REST Service	DM1	RUNNING
DSC	DM1	RUNNING
DSA REST	DM1	RUNNING
DM Agent	DM1	RUNNING
DM Agent	DM2	RUNNING
DM Agent	DM3	RUNNING
REMOTE CLUSTER		
COMPONENT	HOST NAME	STATUS
DM Daemon	DM2	STOPPED
ActiveMQ	DM2	STOPPED
DM Monitoring Service	VP2	STOPPED
DM Sync Monitor	DM2	STOPPED
DM REST Service	DM2	STOPPED
DSC	DM2	STOPPED
DSA REST	DM2	STOPPED
DM Agent	DM2	STOPPED

DM Agent	DM1	RUNNING
DM Agent	DM2	RUNNING
DM Agent	DM3	RUNNING

Cluster Mode: DM Cluster is in 'Active Mode'

When the designated-active Data Mover sync monitor is running and monitoring the synchronization progress from the active to standby repository, the standby Data Mover sync monitor is not in use.

Standby Mode

When failover occurs, Data Mover services run on the designated-standby server and monitored by the standby monitor, the cluster is in standby mode. When checking the status of a standby cluster with the `dmcluster status` command, the following displays:

LOCAL CLUSTER		
COMPONENT	HOST NAME	STATUS
DM Daemon	DM1	STOPPED
ActiveMQ	DM1	STOPPED
DM Monitoring Service	VP1	STOPPED
DM Sync Monitor	DM1	STOPPED
DM REST Service	DM1	STOPPED
DSC	DM1	STOPPED
DSA REST	DM1	STOPPED
DM Agent	DM1	RUNNING
DM Agent	DM2	RUNNING
DM Agent	DM3	RUNNING
REMOTE CLUSTER		
COMPONENT	HOST NAME	STATUS
DM Daemon	DM2	RUNNING
ActiveMQ	DM2	RUNNING
DM Monitoring Service	VP2	RUNNING
DM Sync Monitor	DM2	STOPPED
DM REST Service	DM2	RUNNING
DSC	DM2	RUNNING
DSA REST	DM2	RUNNING
DM Agent	DM1	RUNNING
DM Agent	DM2	RUNNING
DM Agent	DM3	RUNNING

Cluster Mode: DM Cluster is in 'Standby Mode'.

WARNING: Data Mover will not do failover in 'Standby Mode'. Please run 'dmcluster switchback' on primary DM server to go back to 'Active Mode' as soon as possible.

The Data Mover cluster runs in a downgraded state when in standby more. After failover, the sync monitor stops running and the synchronization service from the active to standby repository is disabled. Since an additional failover cannot occur, it is important to switch the cluster back to active mode as soon as possible. For instructions on switching from standby back to active mode, see [Failing Back to the Designated-Active Daemon](#).

Stopping the Monitoring Service

Stopping the active Data Mover daemon, repository, REST, ActiveMQ, or all the active agents for any reason may trigger an automatic failover. If any of these active Data Mover components need to be stopped, do the following:

1. From the designated-active daemon or the active monitor server, run the `dmcluster stopmonitor` command to stop the monitoring service on both monitoring servers.
2. Run the `dmcluster status` command to verify the monitoring service has stopped before stopping any of the running components.

Starting the Monitoring Service

1. From the designated-active or standby daemon server, run the `dmcluster status` command to [verify the failover cluster status](#).
2. Using the `dmcluster status` output, confirm that all active components have started before starting the monitoring service.
3. On the current active monitoring server, based on the cluster status, run `dmcluster startmonitor`.

Note:

If the Data Mover cluster is in standby mode, a failover has been initiated. After failover, you can only restart services on the designated-standby monitor server. For more information about switching the Data Mover cluster back to active mode after a failover, see [Failing Back to the Designated-Active Daemon](#)

Restarting Services

1. From the designated-active or standby daemon server, run `dmcluster status` to [verify the cluster state](#).
2. Log on to the currently active daemon server and run the following:

```
dmcluster setmaster
```

The Data Mover components are restarted on the currently active server and stopped on the standby server. Sync monitor starts when the Data Mover cluster is in active mode.

You can use the `setmaster` command any number of times without modifying the failover properties file on the server.
3. From the active daemon server, run `dmcluster status` to verify all services are running as expected. It may take a few minutes for all services to transition. Run `dmcluster status` as needed until the results are as expected. Monitor progress in `dmFailover.log` on the currently active daemon and monitor servers.

Note:

If the Data Mover cluster is in standby mode, a failover has occurred. When the cluster is in standby mode, you can only restart services on the designated-standby daemon server. For more information about switching the Data Mover cluster back to active mode after a failover, see [Failing Back to the Designated-Active Daemon](#)

Failing Back to the Designated-Active Daemon

After failover, Data Mover does not automatically fail back to the designated-active daemon. To fail back to the designated-active daemon after the root cause of the failover has been identified and fixed on the designated-active components, perform the following:

1. On the designated-active daemon server, run the `dmcluster switchback` command and follow the prompts:

Prompt	Action
Do you need to sync up the two repositories? [y or n]	Enter y to sync up.
Enter 1 to back up the active repository to the standby repository. Enter 2 to back up the standby repository to the active repository. [1 or 2]	Provide the direction of sync up by entering either 1 or 2.

Note:

This synchronizes the Data Mover repositories and re-enables the synchronization service. The Data Mover cluster is restored back to active mode and all the components and associated monitoring services are restarted on the designated-active server.

2. Run `dmcluster status` to verify the designated-active daemon has restored successfully. It may take a few minutes for all services to transition. Run `dmcluster status` as needed until the results are as expected. Monitor progress in `dmFailover.log` on the currently active daemon and monitor servers.

Note:

Do not start the daemon on the designated-active daemon server using the `dmdaemon service script /etc/init.d/dmdaemon start`. This causes the monitoring service to assume that the daemon has incorrectly started on the failed server and shuts it down.

Synchronizing the Active and Standby Repositories

When the Data Mover cluster is running in active mode, the sync monitor keeps the repositories in synchronization. Synchronization keeps components on the standby server with an up-to-date repository ready to take over if a failover occurs.

If Data Mover is configured to [log alerts to server management](#), alerts are triggered when the repositories are no longer in synchronization. Checking `dmSync.log` and Postgres logs help identify the root cause to determine the appropriate corrective action.

Re-synchronize the repositories by performing the following:

1. On the designated-active daemon server, run the `dmcluster switchback` command and follow the prompts:

Prompt	Action
Do you need to sync up the two repositories? [y or n]	Enter y to sync up.
Enter 1 to back up the active repository to the standby repository. Enter 2 to back up the standby repository to the active repository. [1 or 2]	Provide the direction of sync up by entering either 1 or 2.

2. From the designated-active daemon server, run the `dmcluster status` command to verify all services are running in active mode as expected.
It may take a few minutes for all services to transition. Run `dmcluster status` as needed until the results are as expected. Monitor progress in `dmFailover.log` on the currently active daemon and monitor servers.

Fixing Sync Service with High Availability

The automatic failover feature uses the [synchronization service](#), which uses Postgres Logical Replication to synchronize the designated-active and designated-standby repositories.

If the underlying synchronization service is broken in a high-availability environment, use `dmcluster switchback` to re-configure and re-synchronize the repositories.

1. Perform the following, based on the TVI alert received when [server management logging](#) is enabled:

TVI Alert	Action
4604001 or 4604003	These alerts are indicative of a broken sync configuration. <ul style="list-style-type: none"> • Re-configure and re-synchronize the repositories using the steps in Synchronizing the Active and Standby Repositories.
4604002	This alert is triggered when the replication process has had significant delays and the designated-standby repository can no longer keep up with the designated-active repository. <ul style="list-style-type: none"> • Refer to Understanding Replication Lag to troubleshoot networking and performance issues causing the performance delay.

TVI Alert	Action
	<ul style="list-style-type: none"> Reconfigure and resynchronize the repositories using the the steps in Synchronizing the Active and Standby Repositories.

Incremental Copy Jobs With Data Mover Failover

Note:

For information on Data Mover Failover implementation, see, [Fixing Sync Service with High Availability](#)

Previously successful Incremental Copy jobs fails when either designated standby Daemon becomes active during a failover or designated active Daemon is re-promoted to active after a switch back:

Error code: 1186, error message: Incremental Restore is not being processed in the correct order for table 'db.table'.

DSA does not automatically support failover. Here the Active and Standby DSCs are copying to the same target table with different DELTA job history, resulting in potential conflict when either DSC takes over after a failover or switch back.

To work around the issue, start the failed incremental job with "ir_execution_type" parameter set to "full" to do a FULL table copy: `datamove start -job_name my_ir_job -ir_execution_type full`

Usage Notes

Use host names or IP addresses as values in `failover.properties`. The following table describes what happens in certain scenarios when using the automatic failover feature:

Scenario	Result
The default password for <code>dmuser</code> has been changed on any of the servers specified in <code>failover.properties</code> .	SSH setup fails, which prevents the active-standby components from starting correctly.
Invalid host names are specified in <code>failover.properties</code> .	The SSH setup fails.
Config is run as a user other than <code>ROOT</code> .	The SSH setup fails.
The <code>dm.rest.endpoint</code> is not modified in <code>commandline.properties</code> .	The command line is not able to connect to a standby REST server when the active REST server becomes unavailable.
The <code>broker.url</code> is not modified in <code>daemon.properties</code> and <code>agent.properties</code> .	The daemon and agent are not able to connect to a standby JMS broker when the active JMS broker becomes unavailable.

Scenario	Result
The <code>sync.ismaster</code> value in <code>sync.properties</code> is not set correctly on both active and standby sync servers.	The sync monitor service does not start correctly.
The active sync monitor become unavailable when the cluster is running in active mode.	Check the synchronization and Postgres logs on the designated active daemon server to diagnose the root cause of the issue. See Fixing Sync Service with High Availability .
Both daemon servers become unavailable after failover monitoring has been enabled.	The monitoring service detects the active daemon server failure and initiates a failover sequence on the standby daemon server. If it is unable to connect to the standby daemon server, it exits and the components need to be reconfigured and restarted using the <code>dmcluster switchback</code> command from the designated-active daemon server once the servers are up.
The designated-active daemon becomes unavailable and a failover has been initiated. The daemon is restarted again using the <code>dmdaemon service</code> script.	The monitoring service notices two daemons running and stops the daemon on the designated active server. To properly restore the unavailable daemon, use the <code>dmcluster switchback</code> command on the designated active server. For more information, see Failing Back to the Designated-Active Daemon .
The standby daemon is started using the <code>dmdaemon service</code> script when the designated-active daemon is still running.	The monitoring service notices two daemons running and stops the daemon on the designated standby server. Components on the designated standby servers cannot be started when the designated active server is running in active mode. If a permanent change of servers is necessary, reconfigure the Data Mover cluster using the <code>dmcluster config</code> command on the server to become the new designated active server.
The JMS broker on the designated-active daemon server becomes unavailable.	Failover occurs if ActiveMQ cannot be started by the failover service after a multiple attempts.
ActiveMQ on the designated-standby daemon starts.	Failover stops the service automatically. Components on the designated standby servers cannot be started when the designated active server is running in active mode. If a permanent change of servers is necessary, reconfigure the Data Mover cluster using the <code>dmcluster config</code> command on the server to become the new designated-active server.

The Synchronization Service

Use the synchronization service when multiple Data Mover servers are installed and automatic failover is not available or enabled. The synchronization service creates a backup system to use as a temporary active system in case of a daemon failover on the active system.

Note:

The `dmsync` commands used in this section apply to the standalone synchronization service, and are not used during automatic failover. Automatic failover makes use of the synchronization service using `dmccluster` commands from the automatic failover feature.

When using the synchronization service with multiple Data Mover servers, the following terminology applies:

Active server

The Data Mover server where the daemon is running.

Standby server

One or more Data Mover servers where the daemon is not running. Use standby servers as the primary system if a failure on the designated (or primary) active server occurs.

Postgres Logical Replication

Data Mover synchronization services uses Postgres replication under the covers to synchronize data from the active repository to the standby repository. Postgres replication works in the following order:

- A publication is created on the active repository
- A subscription is created on the standby repository
- Data Mover tables are added to the publication
- The publication detects all changes in the Data Mover tables and sends the changes to the standby repository
- The subscription on the standby receives the changes and applies them to the standby repository

The publication and subscription are automatically created when the synchronization service is set up.

Synchronization Commands

Use the Data Mover synchronization service to help you set up the Postgres logical replication. The provided four commands are user-interactive and must be run from the active server, unless otherwise noted.

Command	Description
config	<p>Runs on the active server only and sets up the publication on the active repository and the subscription on the standby repository. Connects the subscription on the standby repository to the publication on the active repository. There are two options when running the <code>config</code> command:</p> <ul style="list-style-type: none"> • Set up a new replication service between the active repository and one or more standby repositories. <ul style="list-style-type: none"> ◦ This is a fresh setup where either there is no existing publication on the active repository or the active repository needs the publication recreated. ◦ This command creates the publication on the active repository and the subscription on the standby repository. It also connects the standby repository to the active repository. • Add additional standby repositories to the existing replication service.

Command	Description
	<ul style="list-style-type: none"> ◦ Run this command when a publication already exists on the active repository. ◦ This command creates a subscription on a standby repository and connects that repository to the active repository.
dropconfig	<p>Runs on the active server only and drops an existing replication service. There are two options when running the dropconfig command:</p> <ul style="list-style-type: none"> • Drop replication on the active repository server. <ul style="list-style-type: none"> ◦ This option drops the replication service completely on the active and standby servers. ◦ Prompts for the standby host name and Postgres user credentials are required. • Drop replication on the standby repository servers. <ul style="list-style-type: none"> ◦ This option drops the subscription on the standby repository servers only. Publication on the active repository server remains untouched.
startmonitor	<p>Runs on the active server only and starts monitoring data replicated between publication and subscription. The following conditions trigger an alert:</p> <ul style="list-style-type: none"> • Publication does not exist on the active repository • Any established standby repository is not active • Data replication is slower than the specified threshold
stopmonitor	Stops the synchronization monitoring service.

Understanding Replication Lag

A slowing in the data replication progress between the active repository and standby repository is called lag. If the slowing exceeds the user-specified lag threshold, TVI alert [4604002](#) is triggered.

The property **sync.data.lagging.threshold** specifies the threshold in the [sync.properties](#) file.

The Postgres transaction process is called *write-ahead logging* (WAL).

The `sync.log` reports the following when the synchronization service is slow:

- `sending_lag`: how many WALs have been generated, but not yet sent to the standby servers.
- `receiving_lag`: WALs in the network that have been sent but not yet written.
- `write_lag`: WALs that have been written but not moved to the permanent storage. If Postgres crashes, these changes are lost.
- `replaying_lag`: WALs that has been moved to the permanent storage but not yet replayed.

The following are possible reasons for system slowing based on the messages in the report:

- `sending_lag`:
 - Active repository performance issue (such as a heavy load)
 - Low throughput of the network between the active and the standby repositories
 - The standby server being offline over a long period of time prior to its starting
- `receiving_lag`:
 - Low throughput of the network between the active and the standby repositories

- Standby repository performance issue (such as a heavy load)
- `write_lag` and `replay_lag`
 - Standby performance issues such as over-utilized storage, stuck recovery process, or a heavy load on standby

sync.properties File

The `sync.properties` file contains all the properties the Data Mover daemon uses, as listed in the following table:

Property	Description	Default Value
<code>master.port</code>	The Postgres port of the active repository that the standby repository uses to connect.	5432
<code>sync.isMaster</code>	Determines if current server is an active or standby server. Options are: <ul style="list-style-type: none"> • <code>true</code> - starts as an active server • <code>false</code> - starts as a standby server 	False
<code>dm.pg.jobstore.production.host</code>	System hostname where the active or standby repository is located.	localhost
<code>sync.monitor.interval=5</code>	Specifies how often the synchronization monitor checks the standby server status.	5 minutes
<code>sync.data.lagging.threshold</code>	Specifies the data lagging threshold that triggers TVI alerts.	2048KB

Configuring the Synchronization Service

When configuring the sync service, do not have running jobs on either the active or standby servers and make sure the repositories are in synchronization. You can synchronize the repositories using one of the following methods:

- Before configuration: Run the `backup_daemon` and `restore_daemon` commands
- During configuration: Let `dmsync config` perform the backup and restore during the configuration process

Note:

The `dmsync config` command does not check if there are running jobs when doing a backup and restore of the system. If this is a concern, run the Data Mover backup and restore commands before configuring the synchronization service.

1. On the active repository server, go to `cd /var/opt/teradata/postgres/data` to change the Postgres configuration.

2. In `postgresql.conf`, change `wal_level` to `logical`.
`wal_level = logical`
3. Restart the active repository:
 - `/etc/init.d/postgresql stop`
 - `/etc/init.d/postgresql start`
4. On the active server, edit the `sync.properties` file and set the `sync.isMaster` property to `true`.
5. On the standby server, edit the `sync.properties` file and set the `sync.isMaster` property to `false`.
6. On the active server, set the `dm.pg.jobstore.production.host` property in the `sync.properties` file as the hostname for the active server.
 Use a name other than `localhost` as the host name.
7. On the active server, configure the sync service:
`/opt/teradata/datamover/sync/nn.nn/dmsync config`
 Where `nn.nn` in the path refers to the version numbers of Data Mover.
8. When prompted, respond to the following:
 - a. Select the **Setup Replication Service** option.
 - b. Provide the standby server name.
 - c. On the standby server, provide the Postgres user credentials.
 - d. Answer `y` or `n` to backup and restore the repository from the active to the standby server.
9. Verify that `/var/opt/teradata/logs/dmSync.log` is configured properly and the sync monitor is running on the active server.
 The monitoring service automatically starts once the configuration process completes successfully.

Dropping the Synchronization Service

Use the `dmsync` command in situations where the synchronization between the active and standby server needs to be deleted.

You have the option to either remove one standby service from the current sync service or remove the active server and all the standby servers from the configuration. The command can only be initiated from the active server.

1. From the active server, go to `cd /opt/teradata/datamover/sync/nn.nn`.
 Where `nn.nn` in the path refers to the version numbers of Data Mover.
2. Run `./dmsync dropconfig`.
3. Select one of the following options to delete either one standby server or the entire replication service:
 - a. Select to delete either one standby server or the entire replication.
 - b. Provide the standby server name.
 - c. Provide the Postgres user credentials on the standby server.

Starting the Synchronization Monitor

The synchronization service starts automatically after the service is successfully configured.

Use the following steps on the active server to manually start the service if necessary.

1. From the active server, go to `cd /opt/teradata/datamover/sync/nn.nn`.
Where *nn.nn* in the path refers to the version numbers of Data Mover.
2. Run `./dmsync startmonitor` to start the synchronization monitoring service.

Stopping the Synchronization Monitor

1. On the active server, go to `cd /opt/teradata/datamover/sync/nn.nn`.
Where *nn.nn* in the path refers to the version numbers of Data Mover.
2. Run `./dmsync stopmonitor` to manually stop the monitoring service.

The Synchronization Service Use Cases

Use Case: Server Synchronization Setup – First Time

1. On the active repository server, go to `cd /var/opt/teradata/postgres/data` to change the Postgres configuration.
2. In `postgres.conf`, change `wal_level` to `wal_level = logical`.
3. Restart the active repository:
`/etc/init.d/postgresql stop`
`/etc/init.d/postgresql start`
4. On the active server, edit the `sync.properties` file and set the `sync.isMaster` property to `true`.
5. On the standby server, edit the `sync.properties` file and set the `sync.isMaster` property to `false`.
6. On the active server, set the `dm.pg.jobstore.production.host` property in the `sync.properties` file as the hostname for the active server.
Use a name other than `localhost` as the host name.
7. On the active server, configure the sync service:
`/opt/teradata/datamover/sync/nn.nn/dmsync config`
Where *nn.nn* in the path refers to the version numbers of Data Mover.
8. When prompted, select the following option:

```
Set up a new replication service between this active repository and one or
more standby repositories
```

9. Respond to the following prompts:

Enter remote standby host name	Enter the standby server name.
Enter the database super user postgres password on standby repository	Enter the Postgres user password on the standby server.

Replication service set up requires that the active and standby repositories are in sync before configuring. Do you need to back up the active repository to the standby repository? [y or n]	<ul style="list-style-type: none"> • Enter Y if standby repository has not yet been synchronized. • Enter N if standby repository has already been synchronized.
Do you need to set up replication for another standby repository? [y or n]:	<ul style="list-style-type: none"> • Enter Y if more than one standby server needs to connect to this active server. • Enter N if only one standby is needed.

Use Case: Add an Extra Standby Server to an Existing Active Server

1. On the standby server, edit the `sync.properties` file and set the `sync.isMaster` property to false.
2. On the active server, configure the sync service:
`/opt/teradata/datamover/sync/nn.nn/dmsync config`
Where `nn.nn` in the path refers to the version numbers of Data Mover.
3. When prompted, select the following option:

Add additional standby repositories to the existing replication service

4. Respond to the following prompts:

Prompt	Description
Enter remote standby host name	Enter the standby server name.
Enter the database super user postgres password on standby repository	Enter the Postgres user password on the standby server.
Replication service set up requires that the active and standby repositories are in sync before configuring. Do you need to back up the active repository to the standby repository? [y or n]	<ul style="list-style-type: none"> • Enter Y if standby repository has not yet been synchronized. • Enter N if standby repository has already been synchronized.
Do you need to set up replication for another standby repository? [y or n]:	<ul style="list-style-type: none"> • Enter Y if more than one standby server needs to connect to this active server. • Enter N if only one standby is needed.

Use Case: Drop One Standby Server from the Synchronization Service

1. On the active server, go to `cd /opt/teradata/datamover/sync/nn.nn`.
Where `nn.nn` in the path refers to the version numbers of Data Mover.
2. Run `./dmsync dropconfig`.
3. At the prompt, select the following option:

Drop replication on standby repository servers

4. Respond to the following prompts:

Prompt	Description
Enter remote standby host name	Enter the standby server name.
Enter the database super user postgres password on standby repository	Enter the Postgres user password on the standby server.
Do you need to drop replication for another standby repository? [y or n]:	<ul style="list-style-type: none"> • Enter y if more than one standby server needs to be dropped. • Enter n if no more standby servers need to be dropped.

Use Case: Drop the Entire Synchronization Service

When a server no longer is needed as an active server, the entire synchronization service can be dropped. Before performing this process, you must know all the servers which are connecting the active server that is being dropped.

1. On the active server, go to `cd /opt/teradata/datamover/sync/nn.nn`.
Where *nn.nn* in the path refers to the version numbers of Data Mover.
2. Run `./dmsync dropconfig`.
3. At the prompt, select the following option:

Drop replicaton on this active repository server

4. Respond to the following prompts:

Prompt	Description
Enter remote standby host name	Enter the standby server name.
Enter the database super user postgres password on standby repository	Enter the Postgres user password on the standby server.
Do you need to drop replication for another standby repository? [y or n]:	<ul style="list-style-type: none"> • Enter y if more than one standby server needs to be dropped. • Enter n if no more standby servers need to be dropped.

Use Case: Switch a Server from Standby to Active

When a standby server is needed to run a job, it must first be dropped from the synchronization service and then switched to active mode.

Currently Data Mover does not block running jobs if a standby has not been dropped from synchronization service and running jobs on a standby server can cause the following when not dropped from the sync service:

- A replication break between the active and this standby server due to a foreign key or primary key issue. The active server will continue to retry sending the data.
 - Jobs may not run successfully.
1. Drop the standby server from the synchronization service using the steps in [Use Case: Drop One Standby Server from the Synchronization Service](#).
 2. Start all Data Mover services on this server such as:
 - `/etc/init.d/tdactivemq start`
 - `/etc/init.d/dmdaemon start`
 - `/etc/init.d/dmrest start`
 - `/etc/init.d/dmagent start`
 3. Run jobs on this newly active server.

Use Case: Recover from a Broken Synchronization Service

When data sent to the standby server cannot be inserted, deleted, or modified successfully, the synchronization service is not working correctly due to either a bad foreign key issue or a duplicate primary key. Perform the following if the synchronization service is not working as expected:

1. Drop this standby server from the synchronization service by following the steps in [Use Case: Drop One Standby Server from the Synchronization Service](#).
2. Add the server back by following the steps in [Use Case: Add an Extra Standby Server to an Existing Active Server](#).

Use Case: TVI Alert 4604001 Is Triggered

The sync monitor service monitors the replication status between the active repository and the standby repository. When the sync service is configured properly, a publication is created in the active repository and a subscription is created in the standby repository.

TVI alert 4604001 means that the publication is not found on the active repository and the sync service is not configured properly.

1. Run the `dmsync config` command and follow the steps in [Use Case: Server Synchronization Setup – First Time](#) to configure the synchronization service.
2. On the active repository, check `dmsync.log` for any errors when the publication was created.

Use Case: TVI Alert 4604002 Is Triggered

Data replication progress between active repository and standby repository may have slowing, if the slowing exceeded user-specified lag threshold, TVI alert 4604002 is triggered. For information about system slowing and alerts, see [Understanding Replication Lag](#).

1. On the standby server, go to `/var/opt/teradata/postgres/data/log/postgresql.log.*` and fix any errors listed that are blocking data replication.
2. If no error found in `postgresql.log.*` and the sequence number in the `dm_sequences` table on the standby is still lagging, check and fix the following:
 - If the lagging threshold is set too low, increase the **`sync.data.lagging.threshold`** in `sync.properties`
 - Check network speed
 - Reconfigure the synchronization service

Use Case: TVI Alert 4604003 Is Triggered

TVI alert 4604003 means that the standby replication service is not active. Possible reasons for this alert are the following:

- There was an issue when replicating data from the active to standby server.
 - The subscription on the standby server is disabled.
1. Select one of the following option:

Option	Action
Re-enable the standby server, if continuing as a standby server	Run <code>dmsync config</code> and select option 2 to add the standby server back to the synchronization service.
Drop the standby server, if server is no longer needed as a standby	Run <code>dmsync dropconfig</code> and select option 2 to drop this standby server from the sync service.

Troubleshooting

Solutions for Job Problems

Problem	Description	Solution
Error When Overwriting Existing Objects	The overwrite option indicates if tables existing on the target are to be overwritten. If the value of the overwrite_existing_objects parameter is false and the table exists on the target, an error message is generated at job creation time: Table XX exists on the target. Change the OverwriteExistingObjects property to true and try again.	If you are creating a new job and get the error message, change the value of the overwrite_existing_objects element to true in create.xml or the overwrite_existing_objects parameter of the create command. You cannot change the option for a job that is already created, therefore create a new job if necessary. If the job was created with the overwrite option specified as false and the table existed on target, but the table was dropped on the target between job creation and run time, the table is not created on the target at job run time.
No Data Mover Agents	If you receive the following error on the command line: Daemon cannot find any agents. Must have at least 1 agent to run a job.	Run <code>/etc/init.d/dmagent status</code> to determine if any Data Mover Agents are currently online. If no Data Mover Agents are online, go to the Teradata server where the Data Mover Agent is installed and do the following: <ol style="list-style-type: none"> 1. Run <code>/etc/init.d/dmagent status</code> to determine if the Data Mover Agent is active. 2. If Data Mover Agent is not active, run <code>/etc/init.d/dmagent</code> to start it.
Tables Not Copied to Target System	If tables are listed in the input XML file, but are not copied to the target Teradata Database system.	You can try any of the following processes to determine the reason: <ul style="list-style-type: none"> • Run the status command with <code>output_level</code> of 4 to view the reason for the failure. • Configure the daemon to send events to Teradata Ecosystem Manager and rerun the job, then review the TMSMEVENT metadata table. • In the create XML file, set the value of the <code>log_level</code> element to 99, then create and start the job. Run the status command with <code>output_level</code> of 4 to view the reason for the failure. • Verify that at least one agent is running by using the <code>list_agents</code> command to list all available agents. • Review the <code>dmDaemon.log</code> file at <code>/var/opt/teradata/datamover/logs</code>. • Review the <code>dmAgent.log</code> file at <code>/var/opt/teradata/datamover/logs</code>.

Problem	Description	Solution
		<ul style="list-style-type: none"> Restart all Teradata® Data Mover services, then run the failed job again. To restart the Teradata® Data Mover ActiveMQ service, change directory to /etc/init.d, then run the following commands: <pre>tdactivemq stop tdactivemq start</pre> <p>To restart the daemon service, change directory to /etc/init.d, then type the following at the command-line:</p> <pre>dmdaemon stop dmdaemon start</pre> <p>To restart the agent service, change directory to /etc/init.d, then type the following at the command-line:</p> <pre>dmagent stop dmagent start</pre>

Logging Levels

You can control the level of logging for a job that you launch through the command-line interface. For the command-line interface, use the **-log_level** parameter of the create and move commands. Teradata recommends using log level 0 or 1 during normal operation.

Value	Command-Line Interface	Description	Details
0	x	Disables logging. This is the default value for the -log_level parameter of the create and move commands.	<ul style="list-style-type: none"> If the job completes successfully, the log is deleted. You cannot view utility logs of successfully completed jobs with the status command. If the job fails, the log is not deleted. You can view the logs of the failed job with the status command. If a DSA job, this defaults to the DSMAIN error logging level. DSMAIN logs are in the /var/opt/teradata/tdtemp/bar directory on the source and target databases.
1	x	Enables detailed logging.	<ul style="list-style-type: none"> You can view the logs of successfully completed or failed jobs with the status command. If a DSA job, this enables the DSMAIN error logging level. DSMAIN logs are in the /var/opt/teradata/tdtemp/bar directory on the source and target databases.
2	x	Enables extra logging for command-line interface jobs.	<ul style="list-style-type: none"> You can view the logs of successfully completed or failed jobs with the status command. If a DSA job, this enables the DSMAIN info logging level. DSMAIN logs are in the /var/opt/teradata/tdtemp/bar directory on the source and target databases.

Value	Command-Line Interface	Description	Details
			<ul style="list-style-type: none"> The DSA job definition is logged in the Data Mover agent log as a JSON.
99	x	<p>Enables maximum logging.</p> <ul style="list-style-type: none"> For TPT API, both operator and data stream logs are sent to <code>/var/opt/teradata/datamover/logs/temp/task XXX YYYY</code> directory, where XXXX is the job id and YYYY is the task ID. For DSA jobs, this enables the DSMAIN DEBUG logging level. DSA logs are under the <code>/var/opt/teradata/tdtemp/bar</code> directory on the source and target databases. 	<ul style="list-style-type: none"> You can view the logs of successfully completed or failed jobs with the <code>status</code> command. Use log level 99 only under the guidance of a Teradata Technical Support Specialist when a job issue needs extremely detailed debug information. If you enable maximum logging, Data Mover creates temporary directories on the Data Mover agent server that runs the task. These directories are not automatically deleted. When you no longer need them, delete the <code>task_XXXX_YYYY</code> directories in <code>/var/opt/teradata/datamover/logs/temp</code>. If a DSA job, the DSA job definition is logged in the Data Mover agent log as a JSON.

Failure Scenarios

The following is an overview of steps to take after a system or system component has failed and been restored. As with any critical data, save the repository data on a routine basis.

Data Mover Server Goes Down

Perform the following if the Data Mover server goes down:

- Use the command `backup_daemon` to back up the Data Mover repository.
- To preserve configuration information and logging information, back up the entire Data Mover installation directory on the Data Mover server. The default installation directory for the Data Mover daemon is `/opt/teradata/datamover/daemon/nn.nn`. The default installation directory for the Data Mover Agent and Command-Line interface is `/opt/teradata/client/nn.nn/datamover`.

Recovery after the Data Mover Server is Restored

Perform the following after the Data Mover server has been restored:

- Use either the `restore_daemon` command or, if running Data Mover version 16.20.25.00 or later, the restore template script generated with the [backup_daemon](#) command to restore the daemon.

Note:

The `restore_daemon` command can only be used if restoring a Data Mover repository that has the same hash algorithm as the Data Mover repository that was backed up by the `backup_daemon` command.

- Copy the backed-up Data Mover installation directory to the system, if files were lost.
- Restart failed jobs that never completed when the Data Mover server failed.

Network Goes down between Data Mover Server and Teradata Database Systems

If the network between the Data Mover server and the source and target Teradata Database systems goes down, stop sending jobs to Data Mover. Doing so prevents a backlog in the daemon job queue.

Recovery after the Network is Restored

Restart failed jobs that never completed when the network failed.

Using Server Management Logging

Server Management allows critical failures, or TVI alerts, to be reported to Teradata immediately. Logging by Server Management is enabled by default, but can be disabled by setting the value of the `logger.useTviLogger` property in `agent.properties` or `daemon.properties` to `false`.

Server Management errors that are reported by Data Mover are listed here by message ID number. All of the errors are critical.

4601001

Synopsis: Daemon port in use.

Meaning: The daemon port that is used for inbound socket connections from agents is in use by some other process.

Probable Cause: Daemon port is in use by some other process.

Recommendations: Change the daemon port to an available port number or terminate any process that is using the existing daemon port.

4601002

Synopsis: Unexpected repository state.

Meaning: The Data Mover repository is in an unexpected state.

Probable Cause: The Data Mover repository server may be down or the repository may not have the correct list of tables.

Recommendations: The Postgres instance being used as the Data Mover repository may not have logons enabled or it may need to be restarted.

The Data Mover Daemon may need to be re-installed to update the repository with the correct list of tables as well.

4601003

Synopsis: Daemon cannot connect to JMS broker.

Meaning: The JMS broker is unreachable from the daemon.

Probable Cause: ActiveMQ service may be down or daemon may have incorrect broker port/url configuration.

Recommendations: Restart ActiveMQ Service. Verify broker port/url on the daemon.

4601004

Synopsis: Daemon cannot connect to agent.

Meaning: Data Mover agents cannot be reached.

Probable Cause: Data Mover agent may be down.

Recommendations: Restart the Data Mover agent.

4601005

Synopsis: Insufficient space in job repository.

Meaning: The Data Mover job repository has run out of space.

Probable Cause: Insufficient space allocation to the job repository.

Recommendations: Check disk space for directory /var/opt/teradata/postgres/data. Allocate more space to this directory or clean up old jobs or delete old log files in directory /var/opt/teradata/postgres/data/log/.

4601006

Synopsis: Daemon cannot write to the specified Event table.

Meaning: The Data Mover Job cannot write events to the specified Event table.

Probable Cause: Event table is not reachable or there is no more room available in the Event table database.

Recommendations: Check if the Event table is accessible from the Data Mover Daemon server. Also check if the Event table database has sufficient space to store new events.

4601007

Synopsis: The daemon repository purge task has timed out multiple times.

Meaning: The Data Mover job repository is not being cleaned properly.

Probable Cause: The database may be locked or overloaded.

Recommendations: Ensure that the Data Mover repository is not locked by another process. Ensure that the Data Mover repository is not on a shared database which is frequently busy with other requests.

4601008

Synopsis: Data Mover Repository exceeded threshold.

Meaning: The row count in one or more repository tables exceeded the threshold limit of 1 million rows.

Probable Cause: Data Mover performance may degrade and failures may occur.

Recommendations: Confirm the Data Mover Purge service is enabled; if the Purge service is already enabled, modify the purge process to purge additional data.

4602001

Synopsis: Agent cannot connect to JMS broker.

Meaning: The JMS broker is unreachable from the agent.

Probable Cause: ActiveMQ service may be down or agent may have incorrect broker port/url configuration.

Recommendations: Restart ActiveMQ Service. Verify broker port/url on agent.

4603001

Synopsis: Data Mover daemon component is unavailable.

Meaning: The Data Mover Monitoring Service cannot detect the Data Mover daemon component.

Probable Cause: The Data Mover daemon or the Data Mover daemon server may be down.

Recommendations: Failover has been initiated to switch DM components to run on standby. Please investigate the root cause of the failure and run "dmcluster switchback" on the active server when the problem is addressed.

4603002

Synopsis: Data Mover repository is unavailable.

Meaning: The Data Mover monitoring service cannot detect the Data Mover repository database.

Probable Cause: The Data Mover repository database may not be running or the Data Mover repository server may be down.

Recommendations: Failover has been initiated to switch DM components to run on standby. Please investigate the root cause of the failure and run "dmcluster switchback" on the active server when the problem is addressed.

4603003

Synopsis: All Data Mover agent components are unavailable.

Meaning: The Data Mover monitoring service cannot detect any running Data Mover agent components.

Probable Cause: All the Data Mover agent components may be down.

Recommendations: Failover has been initiated to switch DM components to run on standby. Please investigate the root cause of the failure and run "dmcluster switchback" on the active server when the problem is addressed.

4603005

Synopsis: Data Mover Sync Monitor is unavailable on active server.

Meaning: The Data Mover Failover Monitoring service cannot detect the Sync Monitor, which tracks the replication progress when DM components are running on active.

Probable Cause: The Sync Monitor may not be running.

Recommendations: Check DM Sync log on active server. If replication service is broken, run 'dmcluster switchback' to reconfigure.

4603009

Synopsis: ActiveMQ component is unavailable.

Meaning: The Data Mover Failover Monitoring Service cannot detect the ActiveMQ component.

Probable Cause: ActiveMQ component may not be running.

Recommendations: Failover has been initiated to switch DM components to run on standby. Please investigate the root cause of the failure and run “dmcluster switchback” on the active server when the problem is addressed.

4603011

Synopsis: Data Mover REST component is unavailable.

Meaning: The Data Mover Monitoring Service cannot detect the Data Mover REST component.

Probable Cause: Data Mover REST component may be down.

Recommendations: Failover has been initiated to switch DM components to run on standby. Please investigate the root cause of the failure and run “dmcluster switchback” on the active server when the problem is addressed.

4603101

Synopsis: Data Mover Daemon component is unavailable.

Meaning: The Data Mover Failover Monitoring Service cannot detect the Data Mover Daemon component.

Probable Cause: The Data Mover Daemon or the Data Mover Daemon server may be down.

Recommendations: No failover will occur as DM components are already running on standby due to previous failover. Please investigate the root cause of the failover and run “dmcluster switchback” on the active server when the problem is addressed.

4603102

Synopsis: Data Mover Repository is unavailable.

Meaning: The Data Mover Failover Monitoring Service cannot detect the Data Mover Repository Database.

Probable Cause: The Data Mover Repository Database may not be running or the the Data Mover Repository server may be down.

Recommendations: No failover will occur as DM components are already running on standby due to previous failover. Please investigate the root cause of the failover and run “dmcluster switchback” on the active server when the problem is addressed.

4603103

Synopsis: All Data Mover Agent components are unavailable.

Meaning: The Data Mover Failover Monitoring Service cannot detect any running Data Mover Agent components.

Probable Cause: All the Data Mover Agent components may be down.

Recommendations: No failover will occur as DM components are already running on standby due to previous failover. Please investigate the root cause of the failover and run “dmcluster switchback” on the active server when the problem is addressed.

4603109

Synopsis: ActiveMQ component is unavailable.

Meaning: The Data Mover Failover Monitoring Service cannot detect the ActiveMQ component.

Probable Cause: ActiveMQ component may not be running.

Recommendations: No failover will occur as DM components are already running on standby due to previous failover. Please investigate the root cause of the failover and run “dmcluster switchback” on the active server when the problem is addressed.

4603111

Synopsis: Data Mover REST component is unavailable.

Meaning: The Data Mover Monitoring Service cannot detect the Data Mover REST component.

Probable Cause: Data Mover REST component may be down.

Recommendations: No failover will occur as DM components are already running on standby due to previous failover. Please investigate the root cause of the failover and run “dmcluster switchback” on the active server when the problem is addressed.

4604001

Synopsis: Data Mover sync monitor service stop running or cannot start.

Meaning: The sync monitor service stopped due to an error.

Probable Cause: The sync monitor service cannot connect to active repository or replication is broken.

Recommendations: Check sync service log for any database connectivity error. If replication is broken, run dmsync config to reconfigure replication service between active and standby repositories.

4604002

Synopsis: Standby repository has too much data lagging.

Meaning: The standby repository data replication is behind specified data lagging threshold.

Probable Cause: The standby repository data replication is too slow or has stopped working.

Recommendations: Check sync service log and postgres log for any error messages.

4604003

Synopsis: Replication slot is not active for a standby or replication data lagging values are null.

Meaning: The standby replication slot active flag value is false or the data lagging values are null.

Probable Cause: The replication between active and standby repository is broken.

Recommendations: Run dmsync config to reconfig replication service between active and standby repository.

Server Management life cycle errors that are reported by Data Mover are listed here by message ID number.

1601001

Synopsis: Data Mover daemon started.

Meaning: The Data Mover daemon component has been started.

Probable Cause: The Data Mover daemon component may have been restarted or the node was rebooted.

Recommendations: None.

1602001

Synopsis: Data Mover agent started.

Meaning: The Data Mover agent component has been started.

Probable Cause: The Data Mover agent component may have been restarted or the node was rebooted.

Recommendations: None.

System Health

The `dmhealthcheck.sh` utility scans the local Data Mover host and checks for any issues within the system. The output includes a table that lists the version and status of the components checked, how the components were checked, and the result of the checks. If an issue is found, the table provides a description of the problem and offers a solution.

The utility only checks the local host. If multiple hosts exist in the Data Mover cluster, the utility must be run on each host.

The health check utility has the following requirements:

- Data Mover versions 15.11 and later.
- Session window size must be greater than 10 characters.
- Bash scripting must be available on the system.
- You must log on as root or have root privileges.

The utility uses BTEQ. Although not required to run the health check script, when BTEQ is not installed, the repository is not checked.

Checking System Health

1. Go to the `/opt/teradata/datamover/support/healthcheck` directory.
2. Run `./dmhealthcheck.sh password` where *password* is the Data Mover repository password.

Creating a Diagnostic Bundle for Support

For Data Mover situations such as job failure, job hanging, or other issues that require an incident report, Teradata includes interactive command-line scripts for collecting necessary job and system information. The resulting diagnostic bundle enables Teradata Customer Support to provide optimum analysis and resolution. Customer support is available around-the-clock, seven days a week through the Global Technical Support Center (GSC). To learn more, go to <https://support.teradata.com>.

The `dmagentsupport.sh` file collects the following information from a server running only the Data Mover agent:

- Data Mover log files from the agent server
- Recent temp and task directories

The `dmagentsupport.sh` script creates a `data-mover-agent-support` output file, which contains the following information:

- `DataMover.agent.properties` files
- List of files from the DataMover components installation directory
- OS, kernel, CPU, memory, and disk space information
- Data Mover and TTU packages rpm information

After the script collects the data, a bundle named `DataMover-$currentdate-$hostname-1.zip` is created in `/var/opt/teradata/datamover/support/incidentnumber`.

If the bundle size is larger than 49 MB, additional `.zip` files are created as follows:

- `DataMover-$currentdate-$hostname-2.zip`
- `DataMover-$currentdate-$hostname-3.zip`

1. Create a support incident including the following settings:

Option	Setting
Product Area	System Management Utilities
Problem Type	Teradata® Data Mover

2. Record the incident number and leave the incident open to attach the diagnostic bundle.

Note:

The interactive script prompts you to enter the incident number and other information related to the issue.

- As the root user, locate the scripts at `/opt/teradata/datamover/support/` for every Data Mover server in your environment, and do the following:

Server Type	Description
Data Mover Server	Run <code>dmsupport.sh</code> to create a diagnostic bundle.
Server Running Only Data Mover Agent	Run <code>dmagentsupport.sh</code> to create a diagnostic bundle.

Be sure to include relevant problem descriptions for troubleshooting as prompted.

The `dmsupport.sh` script collects the following information from the Data Mover log files on the Data Mover multi-purpose server:

- ActiveMQ queue information
- Recent temp and task directories
- DSA information, including:
 - DSC and DSA command line utility logs
 - Installation logs
 - Property files
 - RPM information

The script creates three output files:

Output File	Contents
<code>datamover-job-status</code>	<ul style="list-style-type: none"> Data Mover health information Data Mover and TTU packages rpm information List of total and failed Data Mover jobs List of job steps for failed jobs
<code>datamover-properties</code>	All data Mover properties files, including the following: <ul style="list-style-type: none"> List of files from the Data Mover components installation directory <code>ps aux</code> command output
<code>datamover-server-details</code>	OS, kernel, CPU, memory, and disk space information.

- Update the incident, browse to the resulting `.zip` files, attach the resulting files to the incident, and submit them.
- Contact Teradata Customer Support when the diagnostic bundle is ready for review, and include your incident number for reference.
- [Optional] If you do not want to keep the `.zip` files, delete them from the `/var/opt/teradata/datamover/support/incidentnumber` directory on the Data Mover server.

Note:

To capture jstack and jmap output for tdactivemq, daemon, and agent processes, add the option `-runjvmtools` as in the following examples:

```
dmsupport.sh -runjvmtools
```

```
dmagentsupport.sh -runjvmtools
```

Calling scripts with the jstack and jmap option can impact running jobs. Contact Teradata Services for recommendations before specifying.

User Cannot Choose Data Mover Daemon in the Data Mover Portlet

Problem

When selecting a daemon to connect to through the **Data Mover** portlet, the daemon is not available.

Solution

A daemon must be enabled in Viewpoint and the Viewpoint user must have at least a daemon-level read permission on the daemon for it to be available. Check the **Data Mover Setup** portlet to make sure the current Viewpoint user has the daemon-level read permission set for the specific Data Mover daemon.

Job is Queued Even Though No Other Work is Executing

Problem

When starting a job, the job is immediately queued but the Data Mover portlet or the Data Mover command line interface `list_jobs` command does not show any other running jobs.

Solution

Job- and daemon-level permissions are needed to view running jobs when using the `list_jobs` command. With multiple users on a system, it is possible not all jobs are visible to all users.

Data Mover Command Parameters

About Data Mover Commands

Use the commands by typing them in the command-line or by creating a `parameters.xml` file. Specify the location of the file with `-f parameters.xml`. When using an xml file, make sure the parameters are listed in the same order as they appear in the [Data Mover XML Schemas](#) file.

If the same parameter is defined in `parameters.xml` and also directly in the command-line, a message displays and the value from the command-line takes precedence.

For supported RESTful API command parameters, see [RESTful APIs](#).

Parameter Order

All Data Mover command line interface commands have an associated XML type which uses the naming convention **dmcommandname**. Search the Data Mover XML schema file for the top-level XML type for each command. The top-level tag indicates the associated subtypes to review. The XML schema file lists the parameters in the required order that those parameters must be entered in the `parameters.xml` file. Parameter order is important when passing parameters through an XML file to the Data Mover command line interface. Refer to the Data Mover XML schema at [Data Mover XML Schemas](#) to review the required order of parameters.

Note:

Parameters may be in any order when providing parameters directly on the command line when using the Data Mover command line interface. For example, `datamove status -job_namejob1 -output_level 4` is the same as `datamove status -output_level4 - job_namejob1`.

The following chart shows commands commonly provided in the `parameters.xml` file and their associated XML tags.

Command	Schema Object
create	dmCreate
create_event_table	dmCreateEventTable
edit	dmEdit
modify_event_table	dmModifyEventTable
move	dmCreate
save_configuration	dmSaveConfiguration

Command	Schema Object
start	dmStart

backup_daemon

Purpose

The `backup_daemon` command allows you to backup the Data Mover repository. This can be restored using the `restore_daemon` command.

As of Data Mover 17.05, the `backup_daemon` command uses the `repository_backup.sh` scripts located in the `/opt/teradata/datamover/daemon/nn.nn` directory to back up the daemon configuration from the Data Mover repository. The script uses the PostgreSQL `pg_dump` command for backup.

Note:

Where `nn.nn` in the path refers to the version numbers of Data Mover.

Two files are created in the selected backup directory:

- `backup_restore.log` – contains the log information for the backup and restore
- `datamover-backup.tar.gz` – the backup file

Note:

The `repository_backup.sh` script can be used independently for daemon backup. Avoid backing up the daemon while jobs are running.

Syntax

See [Data Mover XML Schemas](#).

Parameters

See [Parameter Order](#).

backup_target_dir

[Optional] The directory where Data Mover writes the backup files. The location you specify can be:

- A path relative to the `daemon_backup` directory. This is a directory reserved for all backup snapshots.

`daemon_backup` is a sub-directory of the install location for the Data Mover daemon on your system. By default, the directory is `/opt/teradata/datamover/daemon/nn.nn`.

- An absolute path. This is any location on the system for which *dm_user* has write privilege, typically /tmp, /var/opt/teradata/datamover/daemon_backup, and /home. backup_daemon runs under the *dm_user* account.

dm.rest.endpoint

[Optional] Enter a Data Mover REST server URL to overwrite the default value specified in the commandline.properties file in order to connect to a different REST server (and therefore a different daemon) at runtime.

https://dm-server1:1443/datamover

security_password

[Optional] Password for the super user or authorized Viewpoint user.

Example:

53cUr17y

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password_encrypted** is also specified.

security_password_encrypted

[Optional] Encrypted password for the super user.

Example:

052c7aabd14c7770141ac3c0137ab98ae0d3f0f7cddf588981206b010c0c1b2f

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password** is also specified.

security_username

[Optional] User ID of the super user or authorized Viewpoint user. The user ID of the super user is dmc1_admin and cannot be changed.

Required if security management is enabled on the Data Mover daemon.

Usage Notes

- If you enter repos_bu001 as the relative path, Data Mover writes the backup files to /var/opt/teradata/datamover/daemon_backup/repos_bu001. Never use an initial slash when specifying a relative path.
- If you enter /home/myhome/repos_bu001 as the absolute path and if *dm_user* has write privilege for this path, Data Mover writes the backup files to /home/myhome/repos_bu001. Always use an initial slash when specifying an absolute path.

- If you do not specify a directory for the backup, Data Mover creates a directory in the format `dm_hostname_nn.nn.nn_YYYY-MM-DD_HH.mm.ss` under `daemon_backup`. This format is composed of `dm`, followed by the `hostname` of the repository, the Data Mover *version*, then the date and timestamp of the backup. For example, a backup that occurs at 2am on hostname *myhost* and Data Mover version 16.20.23.00 on December 3, 2015 is written to the directory `dm_myhost_16.20.23.00_2015-12-03_02.00.00`.
- The following scenarios result in an error:
 - A job is running when you start the `backup_daemon` command
 - You specify an absolute path, but `dm_user` does not have write privilege for the path

XML File Examples

For the `backup_daemon` command, type `datamove backup_daemon -f parameters.xml`.

In the following example, because `parameters.xml` does not specify the backup directory, the backup is written to a directory under `daemon_backup` relative to the install location for the Data Mover daemon on your system. The directory name is based on the timestamp of the backup.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<dmBackupDaemon xmlns="http://schemas.teradata.com/dataMover/v2009"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://schemas.teradata.com/unity/datamover.xsd">
</dmBackupDaemon>
```

If the backup is successful, the name of the directory that was created for the backup is written to standard output.

In the following example, `parameters.xml` specifies `repos_bu_001` as a backup directory under `daemon_backup` relative to the install location for the Data Mover daemon on your system.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<dmBackupDaemon xmlns="http://schemas.teradata.com/dataMover/v2009"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://schemas.teradata.com/unity/datamover.xsd">
  <backup_target_dir>repos_bu_001</backup_target_dir>
</dmBackupDaemon>
```

If the backup is successful, the name of the directory `repos_bu_001` is written to standard output.

cleanup

Purpose

The `cleanup` command cleans up Teradata DSA, Teradata PT API, and Teradata JDBC tasks from a job that failed or was stopped before completing successfully. The command deletes staging, error, work, and log tables and releases HUT locks, depending on the underlying utility used for the job.

Syntax

See [Data Mover XML Schemas](#).

Parameters

See [Parameter Order](#).

dm.rest.endpoint

[Optional] Enter a Data Mover REST server URL to overwrite the default value specified in the `commandline.properties` file in order to connect to a different REST server (and therefore a different daemon) at runtime.

`https://dm-server1:1443/datamover`

job_name

Name of the job to be cleaned up.

Example:

`12315DFHJKS`

security_password

[Optional] Password for the super user or authorized Viewpoint user.

Example:

`53cUr17y`

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password_encrypted** is also specified.

security_password_encrypted

[Optional] Encrypted password for the super user.

Example:

`052c7aabd14c7770141ac3c0137ab98ae0d3f0f7cddf588981206b010c0c1b2f`

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password** is also specified.

security_username

[Optional] User ID of the super user or authorized Viewpoint user. The user ID of the super user is `dmc1_admin` and cannot be changed.

Required if security management is enabled on the Data Mover daemon.

Usage Notes

The following information is specific to the type of job running cleanup:

- Teradata DSA – a DSA stop command is issued for the running job and DSMAIN stops the job and releases any locks.
- Teradata PT API – existing target log, work, error, and staging tables are dropped.
- Teradata JDBC – existing target staging tables are dropped.

The cleanup command only cleans up job instances that are not currently running. To clean up a currently running job, stop the job and then run cleanup.

XML File Example

For the cleanup command, type `datamove cleanup -f parameters.xml`.

In the following example, the parameters file cleans up database artifacts that are left by the stopped or failed job 12315DFHJKS:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<dmCleanup
  xmlns="http://schemas.teradata.com/dataMover/v2009"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://schemas.teradata.com/unity/datamover.xsd">
  <job_name>12315DFHJKS</job_name>
</dmCleanup>
```

create

The create command creates a job on the daemon using the syntax parameters and object list. A job definition consists of the parameters and object list.

Available parameters for this command may vary depending on the involved systems, the direction of data movement, and whether you are using the CLI or an .xml file. Refer to the parameter descriptions for guidance on the context in which you can use the parameters outlined in a section.

create

Purpose

The create command creates a job on the daemon using the syntax parameters and object list. A job definition consists of the parameters and object list.

Syntax

See [Data Mover XML Schemas](#).

Parameters

See [Parameter Order](#).

data_streams

[Optional] Number of data streams to use between the source and target databases. Applies to jobs that use Teradata DSA and TPT API (to and from Teradata). All other protocols use a single data stream.

Example:

4

The default value is dynamically calculated by Data Mover.

db_client_encryption

[Optional] Set to true if job needs to be encrypted during data transfer.

dm.rest.endpoint

[Optional] Enter a Data Mover REST server URL to overwrite the default value specified in the `commandline.properties` file in order to connect to a different REST server (and therefore a different daemon) at runtime.

`https://dm-server1:1443/datamover`

execute_permission

[Optional] Defines the username[s] and role[s] that have execute permission for the created job.

force_utility

[Optional] Forces the Data Mover daemon to use a specific utility for all copy operations.

Valid Values

- dsa
- jdbc
- tptapi
- tptapi_load
- tptapi_stream
- tptapi_update
- T2T

If this value is not specified, the Data Mover daemon determines which Teradata utility is the best to use for the job.

Note:

Copying data to an older version of the Teradata Database using Teradata DSA is not valid. You cannot use Teradata DSA if the source and target TDPIDs are the same.

Example:

dsa

freeze_job_steps

[Optional] Freezes the job steps so that they are not recreated each time the job is started. Only set to true if the source and target environments do not change after the job is created.

Valid Values

- true - the job steps are not recreated each time the job is started
- false - the job steps are recreated each time the job is started
- unspecified (default) - the value is set to false

Example:

true

job_name

[Optional] Name for this job. The name must be unique and up to 32 characters.

If you do not specify a name, it is automatically generated using the format: *<source tdpid>_<target tdpid>_<date time year>*.

job_priority

[Optional] Specifies the execution priority for the job. Supported values are: HIGH, MEDIUM, and LOW, and UNSPECIFIED. If no value is specified, the default of MEDIUM is used at runtime.

Example:

MEDIUM

job_security

[Optional] Defines the access parameters for the created job.

log_level

[Optional] Log level for log file output.

Valid Values

- 0
- 1

- 2
- 99

Example:

2

The default value is 0.

log_to_event_table

[Optional] Specifies the event table to be used for this job. For more information, see [Using Event Tables](#).

max_agents_per_task

[Optional] Maximum number of Data Mover agents to use in parallel when moving tables or databases.

Example:

4

The default value is dynamically calculated by Data Mover.

netrace

[Optional] CLI netrace parameter. Any value greater than or equal to 0 generates a CLI trace log. Valid CLI value must be provided.

netrace_buf_len

[Optional] CLI netrace_buf_len parameter. Any value greater than or equal to 0 generates a CLI trace log. Valid CLI value must be provided.

online_archive

[Optional] Allows read and write access to the source table(s) while the tables are being copied with Teradata DSA. Updates occur to the source table during the copy, but are not transferred to the target table. After a successful copy, the data contained in the target table matches the data that was in the source table at the beginning of the copy.

Valid Values

Value	Description
True	Enables online archive
False	Disables online archive
Unspecified	Default – the value is set to the value in the Data Mover daemon configuration file

Example:

true

overwrite_existing_objects

[Optional] Job overwrites objects that already exist on the target.

Valid Values

Value	Description
True	Enables overwriting
False	Disables overwriting
Unspecified	Default – the value is set to the value in the Data Mover daemon configuration file

If the parameter is not specified, the value is set to the `overwrite_existing_objects` parameter value in the Data Mover daemon configuration file. If the parameter is specified as true or false, that value takes precedence over the parameter value in the Data Mover daemon configuration file.

Example:

true

owner_name

[Optional] User who created the job.

Example:

owner

Set if the daemon security is off or the user is the super user (`dmc1_admin`); otherwise, the actual user logged on overwrites the value.

read_permission

[Optional] Defines the username[s] and role[s] that have read permission for the created job.

response_timeout

[Optional] Amount of time, in seconds, to wait for response from the Data Mover daemon.

Example:

60

source_account_id

[Optional] Logon account ID for source database.

Note:

Spaces in the account name for the source or target account id causes the job to fail.

source_logon_mechanism

[Optional] Logon mechanism for source system. To log on to a source Teradata Database system, the user must provide at least one of the following:

- `source_user` *and* `source_password`
- `source_logon_mechanism`

Logon mechanisms are not supported for Teradata DSA jobs. Use logon mechanisms only for Teradata PT API and Teradata JDBC jobs. If **-source_logon_mechanism** is specified and **-force_utility** is not used, Teradata PT API is used by default. Specifying **-source_logon_mechanism** with Teradata DSA specified for **-force_utility** results in an error.

Example:

KRB5

source_logon_mechanism_data

[Optional] Additional parameters needed for the logon mechanism of the source system.

Example:

joe@domain1@@mypassword

source_password

[Optional] Source Teradata logon password.

Example:

123456789

Not a valid parameter if **-source_password_encrypted** is also specified. If you do not specify a password for this parameter, the command prompts you to enter it interactively. Input is masked with a set number of asterisks, regardless of the length of the password.

source_password_encrypted

[Optional] Source Teradata encrypted logon password.

Example:

17894cc84b5637a88e36fa37a010e3662d18f64b8ce204bef8d63868ad417810

Not a valid parameter if **-source_password** is also specified.

source_sessions

[Optional] Number of sessions per data stream on the source database.

Example:

4

The default value is dynamically calculated by Data Mover.

source_tdpid

Source Teradata Database.

Example:

Checks

source_user

[Optional] Source Teradata logon id.

Example:

TD_API_user

If you do not specify a logon id for this parameter, the command prompts you to enter it interactively.

Note:

Spaces in the user name for the source or target id causes the job to fail.

source_userid_pool

[Optional] Job pulls the user from the specified credential pool. Available for any job type. Must use the same credential pool as `target_userid_pool` if specifying both parameters in the same job definition.

Example:

POOL -1

table

[Optional] Table to be copied.

Example:

DB1.TABLE

target_account_id

[Optional] Logon account ID for target database.

Note:

Spaces in the account name for the source or target account id causes the job to fail.

target_logon_mechanism

[Optional] Logon mechanism for target system. To log on to a target Teradata Database system, the user must provide at least one of the following:

- `target_user` *and* `target_password`
- `target_logon_mechanism`

Teradata DSA does not support logon mechanisms. Use logon mechanisms only with Teradata PT API and Teradata JDBC jobs. If **-target_logon_mechanism** is specified and **-force_utility** is not used, Teradata PT API is used by default. Specifying **-target_logon_mechanism** with Teradata DSA specified for **-force_utility** results in an error.

Example:

KRB5

target_logon_mechanism_data

[Optional] Additional parameters that are needed for the target system's logon mechanism.

Example:

my@domain2@@mypassword

target_password

[Optional] Target Teradata logon password.

Example:

212133344

Not a valid parameter if **-target_password_encrypted** is also specified. If you do not specify a password for this parameter, the command prompts you to enter it interactively. Input is masked with a set number of asterisks, regardless of the length of the password.

target_password_encrypted

[Optional] Target Teradata encrypted logon password.

Example:

30e458fce484cefef07724653f5046095208f69fcfbf76bf7290b8576192c2fe

Not a valid parameter if **-target_password** is also specified.

target_sessions

[Optional] Number of sessions per data stream on the target database.

Example:

4

The default value is dynamically calculated by Data Mover.

target_tdpid

[Optional] Target Teradata Database.

Example:

Leo

target_user

[Optional] Target Teradata logon id.

Example:

TD_tar_User

If you do not specify a logon id for this parameter, the command prompts you to enter it interactively.

Note:

Spaces in the user name for the source or target id causes the job to fail.

target_userid_pool

[Optional] Job pulls the user from the specified credential pool. Available for any job type. Must use the same credential pool as `source_userid_pool` if specifying both parameters in the same job definition.

Example:

POOL -1

tpt_debug

[Optional] TPT API trace debug log parameter. Any value greater than or equal to 0 generates a TPT API trace log. Valid TPT API value must be provided.

write_permission

[Optional] Defines the username[s] and role[s] that have write permission for the created job.

Usage Notes

Type `datamove create -f parameters.xml` to create a job. The job name is displayed on the screen when the create command completes. Remember the job name to use in other commands, such as the stop and start commands.

The create command does not start the job. Use the start command to start the job, or the edit command to review the job scripts.

XML File Example

For the create command, type `datamove create -f parameters.xml`.

The following example shows a parameters file for the create command.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>

<dmCreate
  xmlns="http://schemas.teradata.com/dataMover/v2009"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://schemas.teradata.com/unity/datamover.xsd">
  <job_name>floyd_dmdev_create</job_name>
  <source_tdpid>floyd</source_tdpid>
  <source_user>dmguest</source_user>
  <source_password>please</source_password>
  <target_tdpid>dmdev</target_tdpid>
  <target_user>dmguest</target_user>
  <target_password>please</target_password>
  <data_streams>5</data_streams>
  <source_sessions>1</source_sessions>
  <target_sessions>1</target_sessions>
  <force_utility>dsa</force_utility>
  <log_level>0</log_level>
  <db_client_encryption>false</db_client_encryption>
  <database selection="unselected">
    <name>dmguest</name>
    <table selection="included">
      <name>test1</name>
      <db_client_encryption>true</db_client_encryption>
    </table>
    <table selection="included">
      <name>test2</name>
    </table>
    <table selection="included">
      <name>test3</name>
    </table>
  </database>
</dmCreate>
```

```

</database>
<query_band>Job=payroll;Userid=aa1000000;Jobsession=1122;</query_band>
<job_security>
  <owner_name>owner</owner_name>
<read_permission>
  <username>read_user1</username>
  <username>read_user2</username>
  <role>read_role1</role>
  <role>read_role2</role>
</read_permission>
<write_permission>
  <username>write_user1</username>
  <username>write_user2</username>
  <role>write_role1</role>
  <role>write_role2</role>
</write_permission>
<execute_permission>
  <username>execute_user1</username>
  <username>execute_user2</username>
  <role>execute_role1</role>
  <role>execute_role2</role>
</execute_permission>
</job_security>
</dmCreate>

```

Create a Cloud Staging Copy Job

Purpose

You need a valid cloud staging area for creating a cloud staging copy Job. To create a cloud staging copy Job define the **force_utility** as DSA and add a **cloud_staging_area** element.

Command: create -job_name DMCS2Job -f create.xml

Syntax

[Data Mover XML Schemas](#)

Parameters

See [Parameter Order](#).

The following examples shows the additional DSA parameters added to the XML.

data_streams

[Optional] Number of data streams to use between the source and target databases. Applies to jobs that use Teradata DSA and TPT API (to and from Teradata). All other protocols use a single data stream.

Example:

4

The default value is dynamically calculated by Data Mover.

db_client_encryption

[Optional] Set to true if job needs to be encrypted during data transfer.

dm.rest.endpoint

[Optional] Enter a Data Mover REST server URL to overwrite the default value specified in the `commandline.properties` file in order to connect to a different REST server (and therefore a different daemon) at runtime.

`https://dm-server1:1443/datamover`

execute_permission

[Optional] Defines the username[s] and role[s] that have execute permission for the created job.

force_utility

[Optional] Forces the Data Mover daemon to use a specific utility for all copy operations.

Valid Values

- dsa
- jdbc
- tptapi
- tptapi_load
- tptapi_stream
- tptapi_update
- T2T

If this value is not specified, the Data Mover daemon determines which Teradata utility is the best to use for the job.

Note:

Copying data to an older version of the Teradata Database using Teradata DSA is not valid. You cannot use Teradata DSA if the source and target TDPIIDs are the same.

Example:

dsa

freeze_job_steps

[Optional] Freezes the job steps so that the steps are not recreated each time the job is started. Only set to `true` if the source and target environments do not change after the job is created.

Valid Values

- `true` - the job steps are not recreated each time the job is started
- `false` - the job steps are recreated each time the job is started
- Unspecified (default) - the value is set to `false`

Example:

true

job_name

[Optional] Name for this job. The name must be unique and up to 32 characters.

If you do not specify a name, it is automatically generated using the format: `<source tdpid>_<target tdpid>_<date time year>`.

job_priority

[Optional] Specifies the execution priority for the job. Supported values are: HIGH, MEDIUM, and LOW, and UNSPECIFIED. If no value is specified, the default of MEDIUM is used at runtime.

Example:

MEDIUM

job_security

[Optional] Defines the access parameters for the created job.

log_level

[Optional] Log level for log file output.

Valid Values

- 0
- 1
- 2
- 99

Example:

2

The default value is 0.

log_to_event_table

[Optional] Specifies the event table to be used for this job. For more information, see [Using Event Tables](#).

cloud_staging_area

[Optional] Specifies the cloud staging area to be used for this job. Cloud staging area is necessary to run a cloud staging copy job.

Note:

If **-cloud_storage_area** is specified and **-force_utility** is not used, default use of Teradata DSA happens.

max_agents_per_task

[Optional] Maximum number of Data Mover agents to use in parallel when moving tables or databases.

Example:

4

The default value is dynamically calculated by Data Mover.

netrace

[Optional] CLI netrace parameter. Any value greater than or equal to 0 generates a CLI trace log. Valid CLI value must be provided.

netrace_buf_len

[Optional] CLI netrace_buf_len parameter. Any value greater than or equal to 0 generates a CLI trace log. Valid CLI value must be provided.

online_archive

[Optional] Allows read and write access to the source table(s) while the tables are being copied with Teradata DSA. Updates occur to the source table during the copy, but are not transferred to the target table. After a successful copy, the data contained in the target table matches the data that was in the source table at the beginning of the copy.

Valid Values

Value	Description
True	Enables online archive
False	Disables online archive

Value	Description
Unspecified	Default – the value is set to the value in the Data Mover daemon configuration file

Example:

true

overwrite_existing_objects

[Optional] Job overwrites objects that already exist on the target.

Valid Values

Value	Description
True	Enables overwriting
False	Disables overwriting
Unspecified	Default – the value is set to the value in the Data Mover daemon configuration file

If the parameter is not specified, the value is set to the `overwrite_existing_objects` parameter value in the Data Mover daemon configuration file. If the parameter is specified as true or false, that value takes precedence over the parameter value in the Data Mover daemon configuration file.

Example:

true

owner_name

[Optional] User who created the job.

Example:

owner

Set if the daemon security is off or the user is the super user (`dmc1_admin`); otherwise, the actual user logged on overwrites the value.

read_permission

[Optional] Defines the username[s] and role[s] that have read permission for the created job.

response_timeout

[Optional] Amount of time, in seconds, to wait for response from the Data Mover daemon.

Example:

60

source_account_id

[Optional] Logon account ID for source database.

Note:

Spaces in the account name for the source or target account id causes the job to fail.

source_logon_mechanism

[Optional] Logon mechanism for source system. To log on to a source Teradata Database system, the user must provide at least one of the following:

- `source_user` *and* `source_password`
- `source_logon_mechanism`

Logon mechanisms are not supported for Teradata DSA jobs. Use logon mechanisms only for Teradata PT API and Teradata JDBC jobs. If **-source_logon_mechanism** is specified and **-force_utility** is not used, Teradata PT API is used by default. Specifying **-source_logon_mechanism** with Teradata DSA specified for **-force_utility** results in an error.

Example:

KRB5

source_logon_mechanism_data

[Optional] Additional parameters needed for the logon mechanism of the source system.

Example:

joe@domain1@@mypassword

source_password

[Optional] Source Teradata logon password.

Example:

123456789

Not a valid parameter if **-source_password_encrypted** is also specified. If you do not specify a password for this parameter, the command prompts you to enter it interactively. Input is masked with a set number of asterisks, regardless of the length of the password.

source_password_encrypted

[Optional] Source Teradata encrypted logon password.

Example:

17894cc84b5637a88e36fa37a010e3662d18f64b8ce204bef8d63868ad417810

Not a valid parameter if **-source_password** is also specified.

source_sessions

[Optional] Number of sessions per data stream on the source database.

Example:

4

The default value is dynamically calculated by Data Mover.

source_tdpid

Source Teradata Database.

Example:

Checks

source_user

[Optional] Source Teradata logon id.

Example:

TD_API_user

If you do not specify a logon id for this parameter, the command prompts you to enter it interactively.

Note:

Spaces in the user name for the source or target id causes the job to fail.

source_userid_pool

[Optional] Job pulls the user from the specified credential pool. Available for any job type. Must use the same credential pool as **target_userid_pool** if specifying both parameters in the same job definition.

Example:

POOL-1

table

[Optional] Table to be copied.

Example:

DB1.TABLE

target_account_id

[Optional] Logon account ID for target database.

Note:

Spaces in the account name for the source or target account id causes the job to fail.

target_logon_mechanism

[Optional] Logon mechanism for target system. To log on to a target Teradata Database system, the user must provide at least one of the following:

- `target_user` *and* `target_password`
- `target_logon_mechanism`

Teradata DSA does not support logon mechanisms. Use logon mechanisms only with Teradata PT API and Teradata JDBC jobs. If **-target_logon_mechanism** is specified and **-force_utility** is not used, Teradata PT API is used by default. Specifying **-target_logon_mechanism** with Teradata DSA specified for **-force_utility** results in an error.

Example:

KRB5

target_logon_mechanism_data

[Optional] Additional parameters that are needed for the target system's logon mechanism.

Example:

my@domain2@@mypassword

target_password

[Optional] Target Teradata logon password.

Example:

212133344

Not a valid parameter if **-target_password_encrypted** is also specified. If you do not specify a password for this parameter, the command prompts you to enter it interactively. Input is masked with a set number of asterisks, regardless of the length of the password.

target_password_encrypted

[Optional] Target Teradata encrypted logon password.

Example:

30e458fce484cefef07724653f5046095208f69fcfbf76bf7290b8576192c2fe

Not a valid parameter if **-target_password** is also specified.

target_sessions

[Optional] Number of sessions per data stream on the target database.

Example:

4

The default value is dynamically calculated by Data Mover.

target_tdpid

[Optional] Target Teradata Database.

Example:

Leo

target_user

[Optional] Target Teradata logon id.

Example:

TD_tar_User

If you do not specify a logon id for this parameter, the command prompts you to enter it interactively.

Note:

Spaces in the user name for the source or target id causes the job to fail.

target_userid_pool

[Optional] Job pulls the user from the specified credential pool. Available for any job type. Must use the same credential pool as `source_userid_pool` if specifying both parameters in the same job definition.

Example:

POOL -1

tpt_debug

[Optional] TPT API trace debug log parameter. Any value greater than or equal to 0 generates a TPT API trace log. Valid TPT API value must be provided.

write_permission

[Optional] Defines the username[s] and role[s] that have write permission for the created job.

Usage Notes

Type `datamove create -f create.xml` to create a job. The job name is displayed on the screen when the create command completes.

The create command does not start the job. Use the start command to start the job, or the edit command to review the job scripts.

XML File Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<dmCreate xmlns="http://schemas.teradata.com/dataMover/v2009"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://schemas.teradata.com/dataMover/v2009/
DataMover.xsd">
  <source_tdpid>sourceSys</source_tdpid>
  <source_user>source_user</source_user>
  <source_password>source_password</source_password>
  <target_tdpid>targetSys</target_tdpid>
  <target_user>target_user</target_user>
  <target_password>target_password</target_password>
  <data_streams>1</data_streams>
  <source_sessions>1</source_sessions>
  <target_sessions>1</target_sessions>
  <freeze_job_steps>FALSE</freeze_job_steps>
  <force_utility>DSA</force_utility>
  <log_level>99</log_level>
  <cloud_staging_area>
    <name>csareaname</name>
  </cloud_staging_area>
  <dsa_options>
    <target_group_name>my_target_group</target_group_name>
    <parallel_builds>1</parallel_builds>
  </dsa_options>
  <database selection="unselected">
    <name>db1</name>
    <table selection="included">
      <name>tb1</name>
    </table>
  </database>
</dmCreate>
```

create_event_table

Purpose

The `create_event_table` command creates a table called `TMSMEvent` in the specified database.

Syntax

See [Data Mover XML Schemas](#).

Parameters

See [Parameter Order](#).

dm.rest.endpoint

[Optional] Enter a Data Mover REST server URL to overwrite the default value specified in the `commandline.properties` file in order to connect to a different REST server (and therefore a different daemon) at runtime.

`https://dm-server1:1443/datamover`

event_database

Name of the database on the Teradata server to create the event table.

Example:

`db1`

event_table_name

A unique label for the particular `TMSMEvent` table instance with a maximum of 128 characters. This label is used when performing an action, such as modifying or deleting an event table.

Example:

`event1`

security_password

[Optional] Password for the super user or authorized Viewpoint user.

Example:

`53cUr17y`

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **`-security_password_encrypted`** is also specified.

security_password_encrypted

[Optional] Encrypted password for the super user.

Example:

052c7aabd14c7770141ac3c0137ab98ae0d3f0f7cddf588981206b010c0c1b2f

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password** is also specified.

security_username

[Optional] User ID of the super user or authorized Viewpoint user. The user ID of the super user is `dmc1_admin` and cannot be changed.

Required if security management is enabled on the Data Mover daemon.

system

Name of the Teradata system to install the event table. This can be the IP address or system alias.

Example:

`tdsys1`

user_name

Existing Teradata user for the Teradata system. This user creates the event table and inserts events into it. Data Mover stores the user information. The user must have CREATE and INSERT permissions on the specified event_database.

Example:

`dmuser1`

user_password

Password for the Teradata user associated with the user_name.

Example:

`dmuser1`

use_existing_event_table

If an event table already exists and this is set to true, Data Mover re-uses the existing table rather than create a new one.

Example:

`false`

XML File Example

For the `create_event_table` command, type `datamove create_event_table -f parameters.xml`.

The following example is a parameters file for the `create_event_table` command.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<dmCreateEventTable xmlns="http://schemas.teradata.com/dataMover/v2009"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schemas.teradata.com/unity/datamover.xsd">
    <event_table_name>eventTable1</event_table_name>
    <system>tdsys1</system>
    <user_name>tduser</user_name>
    <user_password>tdpass</user_password>
    <event_database>eventdb</event_database>
    <use_existing_event_table>false</use_existing_event_table>
</dmCreateEventTable>
```

create_cloud_staging

Purpose

The `create_cloud_staging` allows you to create a cloud staging area that you can use when creating a Data Mover job using the Cloud Staging Copy Service. There are two options for creating a cloud staging area:

- Provide Source and Target system Target Groups that have already been defined in DSC and paired in a Target Group Map
- Provide AWS S3 information and allow Data Mover to define the Target Groups and Map

Syntax

[Data Mover XML Schemas](#)

Parameters

See [Parameter Order](#).

dm.rest.endpoint

[Optional] Enter a Data Mover REST server URL to overwrite the default value specified in the `commandline.properties` file in order to connect to a different REST server (and therefore a different daemon) at runtime.

`https://dm-server1:1443/datamover`

response_timeout

[Optional] Amount of time, in seconds, to wait for response from the Data Mover daemon.

Example:

`60`

security_password

[Optional] Password for the super user or authorized Viewpoint user.

Example:

53cUr17y

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password_encrypted** is also specified.

security_password_encrypted

[Optional] Encrypted password for the super user.

Example:

052c7aabd14c7770141ac3c0137ab98ae0d3f0f7cddf588981206b010c0c1b2f

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password** is also specified.

security_username

[Optional] User ID of the super user or authorized Viewpoint user. The user ID of the super user is dmc1_admin and cannot be changed.

Required if security management is enabled on the Data Mover daemon.

Usage Notes

Type `datamove create_cloud_staging -f cloudstagingarea.xml` to create a cloud staging area. When the command is complete a message appears stating that the cloud staging area successfully created.

XML File Example

The following example shows creating a cloud staging area with predefined target groups and target group mappings.

```
<?xml version='1.0' encoding='UTF-8'?>
<dmCreateCloudStaging xmlns="http://schemas.teradata.com/dataMover/v2009"
xsi:schemaLocation="http://schemas.teradata.com/unity/DataMover.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <name>stagingarea</name>
  <storage_type>S3</storage_type>
  <source_target_pairs>
    <source_target_pair>
      <source_system>sourceSystem</source_system>
    <source_system_target_group>sourceSystem_stagingarea_tg</
source_system_target_group>
```

```

<target_system>targetSystem/target_system>
<target_system_target_group>targetSystem_stagingarea_tg</
target_system_target_group>
    </source_target_pair>
</source_target_pairs>
</dmCreateCloudStaging>

```

The following example shows creating a cloud staging area where Data Mover configures DSC for AWS S3, and automatically configures target groups and target group mappings.

```

<?xml version='1.0' encoding='UTF-8'?>
<dmCreateCloudStaging xmlns="http://schemas.teradata.com/dataMover/v2009"
xsi:schemaLocation="http://schemas.teradata.com/unity/DataMover.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <name>csa1</name>
    <storage_type>S3</storage_type>
    <s3_properties>
        <access_key_id>ABCDEFGH</access_key_id>
        <secret_access_key>AbcDEfgHIjklmNop123/456qRstUVwXyZ</secret_access_key>
        <buckets_by_regions>
            <buckets_by_region>
                <region>us-west-2</region>
                <buckets>
                    <bucket>
                        <bucket_name>my-s3-bucket</bucket_name>
                        <prefix_list>
                            <prefix>
                                <prefix_name>backup</prefix_name>
                                <storage_devices>100</storage_devices>
                            </prefix>
                        </prefix_list>
                    </bucket>
                </buckets>
            </buckets_by_region>
        </buckets_by_regions>
    </s3_properties>
    <source_target_pairs>
        <source_target_pair>
            <source_system>sourceSystem1</source_system>
            <target_system>targetSystem1</target_system>
        </source_target_pair>
    </source_target_pairs>
</dmCreateCloudStaging>

```

delete_event_table

Purpose

The delete_event_table command enables you to delete an existing Event table.

Syntax

See [Data Mover XML Schemas](#).

Parameter

See [Parameter Order](#).

dm.rest.endpoint

[Optional] Enter a Data Mover REST server URL to overwrite the default value specified in the commandline.properties file in order to connect to a different REST server (and therefore a different daemon) at runtime.

`https://dm-server1:1443/datamover`

event_table_name

Label that uniquely identifies the installation of the Event table.

Example:

`event1`

security_password

[Optional] Password for the super user or authorized Viewpoint user.

Example:

`53cUr17y`

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password_encrypted** is also specified.

security_password_encrypted

[Optional] Encrypted password for the super user.

Example:

`052c7aabd14c7770141ac3c0137ab98ae0d3f0f7cddf588981206b010c0c1b2f`

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password** is also specified.

security_username

[Optional] User ID of the super user or authorized Viewpoint user. The user ID of the super user is `dmcl_admin` and cannot be changed.

Required if security management is enabled on the Data Mover daemon.

XML File Example

For the `delete_event_table` command, type `datamove delete_event_table -f parameters.xml`.

The following example is a parameters file for the `delete_event_table` command.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<dmDeleteEventTable xmlns="http://schemas.teradata.com/dataMover/v2009"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schemas.teradata.com/unity/datamover.xsd">
  <event_table_name>eventTable1</event_table_name>
</dmDeleteEventTable>
```

Data Mover XML Schema

The XML schema provides a blueprint for the format of the `parameters.xml` file used to provide parameter values when running Data Mover commands. The XML schema for the Data Mover Command Line Interface can be found at [Data Mover XML Schemas](#).

delete_cloud_staging**Purpose**

The `delete_cloud_staging` allows you to delete a cloud staging area.

Syntax

[Data Mover XML Schemas](#)

Parameters

See [Parameter Order](#).

dm.rest.endpoint

[Optional] Enter a Data Mover REST server URL to overwrite the default value specified in the `commandline.properties` file in order to connect to a different REST server (and therefore a different daemon) at runtime.

`https://dm-server1:1443/datamover`

response_timeout

[Optional] Amount of time, in seconds, to wait for response from the Data Mover daemon.

Example:

60

security_password

[Optional] Password for the super user or authorized Viewpoint user.

Example:

53cUr17y

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password_encrypted** is also specified.

security_password_encrypted

[Optional] Encrypted password for the super user.

Example:

052c7aabd14c7770141ac3c0137ab98ae0d3f0f7cddf588981206b010c0c1b2f

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password** is also specified.

security_username

[Optional] User ID of the super user or authorized Viewpoint user. The user ID of the super user is `dmcl_admin` and cannot be changed.

Required if security management is enabled on the Data Mover daemon.

Usage Notes

You can use `delete_cloud_staging` command with the command line parameter or an XML file. Type `datamove delete_cloud_staging -name my_staging_area` to delete a cloud staging area using the command line parameter. Type `datamove delete_cloud_staging -f deletecloudstaging.xml` to delete a cloud staging area using an XML file.

XML File Example

```
<?xml version='1.0' encoding='UTF-8'?>
<dmDeleteCloudStaging xmlns="http://schemas.teradata.com/dataMover/v2009"
xsi:schemaLocation="http://schemas.teradata.com/unity/DataMover.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <name>my_staging_area</name>
</dmDeleteCloudStaging>
```

delete_job

Purpose

The `delete_job` command deletes the specified job from the daemon. If an instance of a job is specified, that job instance is deleted. Before the job instance is deleted, a cleanup is performed, if required. If a job definition is specified, all job instances are deleted, but not the job definition itself. Before all job instances are deleted, a cleanup is performed on the last job instance, if required. There is also a parameter, *all*, to delete the job definition and all job instances.

Syntax

See [Data Mover XML Schemas](#).

Parameters

See [Parameter Order](#).

all

[Optional] Deletes job definition and all job instances.

dm.rest.endpoint

[Optional] Enter a Data Mover REST server URL to overwrite the default value specified in the `commandline.properties` file in order to connect to a different REST server (and therefore a different daemon) at runtime.

`https://dm-server1:1443/datamover`

job_name

Name of the job to be deleted.

Example:

`12315DFHJKS`

security_password

[Optional] Password for the super user or authorized Viewpoint user.

Example:

`53cUr17y`

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password_encrypted** is also specified.

security_password_encrypted

[Optional] Encrypted password for the super user.

Example:

052c7aabd14c7770141ac3c0137ab98ae0d3f0f7cddf588981206b010c0c1b2f

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password** is also specified.

security_username

[Optional] User ID of the super user or authorized Viewpoint user. The user ID of the super user is `dmc1_admin` and cannot be changed.

Required if security management is enabled on the Data Mover daemon.

skip_prompt

[Optional] Skips the prompt that confirms deletion.

XML File Example

For the `delete_job` command, type `datamove delete_job -f parameters.xml`.

In the following example, the parameters file deletes the job definition and all job instances for `dmdev_to_floyd-Tue_Aug_25_06:15:56_EDT_2009` and skips the prompt that confirms deletion.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<dmDeleteJob
  xmlns="http://schemas.teradata.com/dataMover/v2009"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://schemas.teradata.com/unity/datamover.xsd">
    <job_name>dmdev_to_floyd-Tue_Aug_25_06:15:56_EDT_2009</job_name>
    <all>true</all>
    <skip_prompt>true</skip_prompt>
</dmDeleteJob>
```

edit

The `edit` command modifies an existing job definition. All previous job executions will be linked with the new job definition.

Available parameters for this command may vary depending on the involved systems, the direction of data movement, and whether you are using the CLI or an `.xml` file. Refer to the parameter descriptions for guidance on the context in which you can use the parameters outlined in a section.

edit

Purpose

The edit command modifies an existing job definition. All previous job executions will be linked with the new job definition.

Syntax

See [Data Mover XML Schemas](#).

Parameters

See [Parameter Order](#).

data_streams

[Optional] Number of data streams to use between the source and target databases. Applies to jobs that use Teradata DSA and TPT API (to and from Teradata). All other protocols use a single data stream.

Example:

4

The default value is dynamically calculated by Data Mover.

db_client_encryption

[Optional] Set to true if job needs to be encrypted during data transfer.

force_utility

[Optional] Forces the Data Mover daemon to use a specific utility for all copy operations.

Valid Values

- dsa
- jdbc
- tptapi
- tptapi_load
- tptapi_stream
- tptapi_update
- T2T

If this value is not specified, the Data Mover daemon determines which Teradata utility is the best to use for the job.

Note:

Copying data to an older version of the Teradata Database using Teradata DSA is not valid. You cannot use Teradata DSA if the source and target TDPIDs are the same.

Example:

dsa

job_name

Name of the job to be edited

Example:

12315DFHJKS

job_priority

[Optional] Specifies the execution priority for the job. Supported values are: HIGH, MEDIUM, and LOW, and UNSPECIFIED. If no value is specified, the default of MEDIUM is used at runtime.

Example:

MEDIUM

log_level

[Optional] Log level for log file output.

Valid Values

- 0
- 1
- 2
- 99

Example:

2

The default value is 0.

max_agents_per_task

[Optional] Maximum number of Data Mover agents to use in parallel when moving tables or databases.

Example:

4

The default value is dynamically calculated by Data Mover.

netrace

[Optional] CLI netrace parameter. Any value greater than or equal to 0 generates a CLI trace log. Valid CLI value must be provided.

netrace_buf_len

[Optional] CLI netrace_buf_len parameter. Any value greater than or equal to 0 generates a CLI trace log. Valid CLI value must be provided.

online_archive

[Optional] Allows read and write access to the source table(s) while the tables are being copied with Teradata DSA. Updates occur to the source table during the copy, but are not transferred to the target table. After a successful copy, the data contained in the target table matches the data that was in the source table at the beginning of the copy.

Valid Values

Value	Description
True	Enables online archive
False	Disables online archive
Unspecified	Default – the value is set to the value in the Data Mover daemon configuration file

Example:

true

overwrite_existing_objects

[Optional] Job overwrites objects that already exist on the target.

Valid Values

Value	Description
True	Enables overwriting
False	Disables overwriting
Unspecified	Default – the value is set to the value in the Data Mover daemon configuration file

If the parameter is not specified, the value is set to the `overwrite_existing_objects` parameter value in the Data Mover daemon configuration file. If the parameter is specified as true or false, that value takes precedence over the parameter value in the Data Mover daemon configuration file.

Example:

true

query_band

[Optional] A semicolon-separated set of name-value pairs that uniquely identifies Teradata sessions or transactions for the source and target. To use a query band to identify the job payroll , the user ID aa100000 , and job session number 1122, define the query band as follows:

Example:

Job=payroll;Userid=aa100000;Jobsession=1122;

This parameter must be added as the last parameter in the XML job definition. See [About Query Band](#) for detailed syntax rules when defining a query band.

response_timeout

[Optional] Amount of time, in seconds, to wait for response from the Data Mover daemon.

Example:

60

security_password

[Optional] Password for the super user or authorized Viewpoint user.

Example:

53cUr17y

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password_encrypted** is also specified.

security_password_encrypted

[Optional] Encrypted password for the super user.

Example:

052c7aabd14c7770141ac3c0137ab98ae0d3f0f7cddf588981206b010c0c1b2f

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password** is also specified.

security_username

[Optional] User ID of the super user or authorized Viewpoint user. The user ID of the super user is dmc1_admin and cannot be changed.

Required if security management is enabled on the Data Mover daemon.

source_account_id

[Optional] Logon account ID for source database.

Note:

Spaces in the account name for the source or target account id causes the job to fail.

source_logon_mechanism

[Optional] Logon mechanism for source system. To log on to a source Teradata Database system, the user must provide at least one of the following:

- `source_user` *and* `source_password`
- `source_logon_mechanism`

Logon mechanisms are not supported for Teradata DSA jobs. Use logon mechanisms only for Teradata PT API and Teradata JDBC jobs. If **-source_logon_mechanism** is specified and **-force_utility** is not used, Teradata PT API is used by default. Specifying **-source_logon_mechanism** with Teradata DSA specified for **-force_utility** results in an error.

Example:

KRB5

source_logon_mechanism_data

[Optional] Additional parameters needed for the logon mechanism of the source system.

Example:

joe@domain1@@mypassword

source_password

[Optional] Source Teradata logon password.

Example:

123456789

Not a valid parameter if **source_password_encrypted** is also specified.

source_password_encrypted

[Optional] Source Teradata encrypted logon password.

Example:

17894cc84b5637a88e36fa37a010e3662d18f64b8ce204bef8d63868ad417810

Not a valid parameter if **-source_password** is also specified.

source_sessions

[Optional] Number of sessions per data stream on the source database.

Example:

4

The default value is dynamically calculated by Data Mover.

source_tdpid

[Optional] Source Teradata Database.

Example:

Checks

source_user

[Optional] Source Teradata logon id.

Example:

TD_API_user

source_userid_pool

[Optional] Job pulls the user from the specified credential pool. Available for any job type. Must use the same credential pool as `target_userid_pool` if specifying both parameters in the same job definition.

Example:

POOL-1

target_account_id

[Optional] Logon account ID for target database.

Note:

Spaces in the account name for the source or target account id causes the job to fail.

target_logon_mechanism

[Optional] Logon mechanism for target system. To log on to a target Teradata Database system, the user must provide at least one of the following:

- `target_user` *and* `target_password`
- `target_logon_mechanism`

Teradata DSA does not support logon mechanisms. Use logon mechanisms only with Teradata PT API and Teradata JDBC jobs. If **-target_logon_mechanism** is specified and **-force_utility** is not used, Teradata PT API is used by default. Specifying

-target_logon_mechanism with Teradata DSA specified for **-force_utility** results in an error.

Example:

KRB5

target_logon_mechanism_data

[Optional] Additional parameters that are needed for the target system's logon mechanism.

Example:

my@domain2@@mypassword

target_password

[Optional] Target Teradata logon password.

Example:

212133344

Not a valid parameter if **-target_password_encrypted** is also specified.

target_password_encrypted

[Optional] Target Teradata encrypted logon password.

Example:

30e458fce484cefef07724653f5046095208f69fcfbf76bf7290b8576192c2fe

Not a valid parameter if **-target_password** is also specified.

target_sessions

[Optional] Number of sessions per data stream on the target database.

Example:

4

The default value is dynamically calculated by Data Mover.

target_tdpid

[Optional] Target Teradata Database.

Example:

Leo

target_user

[Optional] Target Teradata logon id.

Example:

TD_tar_user

target_userid_pool

[Optional] Job pulls the user from the specified credential pool. Available for any job type. Must use the same credential pool as source_userid_pool if specifying both parameters in the same job definition.

Example:

POOL-1

tpt_debug

[Optional] TPT API trace debug log parameter. Any value greater than or equal to 0 generates a TPT API trace log. Valid TPT API value must be provided.

uowid

[Optional] Alternate ID or name for the batch of work associated with the job. If you provide a value for this parameter, Data Mover reports this value as the unit of work ID when sending events to Teradata Ecosystem Manager or to its internal TMSMEVENT table. If you do not specify this parameter, Data Mover uses a default value as the unit of work ID when sending events to Teradata Ecosystem Manager or to its internal TMSMEVENT table. The default value for the unit of work ID is composed of the job execution name and the current timestamp. For example, if you want to define the origins of a query source the job execution name is sales_table, the default value of the unit of work ID is sales_table-20211110122330

Example:

sales_tables_start

Usage Notes

If security is enabled and **job_security** is specified in the modified XML to change job owner, the user must be dcm1_admin; if security is enabled and **job_security** is specified in the modified XML to change job permission, the user must be dcm1_admin or the job owner, and the user must provide all the permissions, not just the modified permissions

Make sure the list of objects in the XML file contains all objects to be moved, not just the objects with name modifications. If an object specified in the original job is not included in the updated list, it is not removed from the new job definition. If no objects are listed, Data Mover assumes there are no updates to original object list.

XML File Example

For the edit command, type `datamove edit -f parameters.xml`.

The following example shows a parameters file for the edit command.

```

<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<dmEdit xmlns="http://schemas.teradata.com/dataMover/v2009"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schemas.teradata.com/unity/datamover.xsd">
  <!-- source and target ip not provided, they will be retrieved from original
  job def -->
    <job_name>job_name</job_name>
    <source_user>dm12066</source_user>
    <source_password>dbc</source_password>
    <target_user>dm12066</target_user>
    <target_password>dbc</target_password>
  <!-- The following are the object which are different from original job def -->
  <database selection="unselected">
    <name>dm12066</name>
    <table selection="included">
      <name>fmt_inf</name>
      <validate_row_count>ALL</validate_row_count>
      <compare_ddl>true</compare_ddl>
    </table>
    <table selection="included">
      <name>NOPITab</name>
      <validate_row_count>ALL</validate_row_count>
      <compare_ddl>true</compare_ddl>
    </table>
  </database>
  <views>
    <view selection="included" copyData="true">
      <name>arrayTabView</name>
      <database>dm12066</database>
      <validate_row_count>partial</validate_row_count>
      <compare_ddl>true</compare_ddl>
      <sql_where_clause><![CDATA[ where c1 = 2]]></sql_where_clause>
      <key_columns>
        <key_column>c1</key_column>
      </key_columns>
    </view>
  </views>
</dmEdit>

```

edit_cloud_staging

Purpose

The `edit_cloud_staging` allows you to edit an existing cloud staging area. Limitations to what you can edit in an existing cloud staging area are:

- Access Key ID (Only if initially provided when creating the cloud staging area.)
- Secret Access Key (Only if initially provided when creating the cloud staging area.)
- Source Target Pairs.

Syntax

[Data Mover XML Schemas](#)

Parameters

See [Parameter Order](#).

dm.rest.endpoint

[Optional] Enter a Data Mover REST server URL to overwrite the default value specified in the `commandline.properties` file in order to connect to a different REST server (and therefore a different daemon) at runtime.

`https://dm-server1:1443/datamover`

response_timeout

[Optional] Amount of time, in seconds, to wait for response from the Data Mover daemon.

Example:

`60`

security_password

[Optional] Password for the super user or authorized Viewpoint user.

Example:

`53cUr17y`

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password_encrypted** is also specified.

security_password_encrypted

[Optional] Encrypted password for the super user.

Example:

`052c7aabd14c7770141ac3c0137ab98ae0d3f0f7cddf588981206b010c0c1b2f`

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password** is also specified.

security_username

[Optional] User ID of the super user or authorized Viewpoint user. The user ID of the super user is `dmc1_admin` and cannot be changed.

Required if security management is enabled on the Data Mover daemon.

Usage Notes

Type `datamove edit_cloud_staging -f editcloudstagingarea.xml` to edit a cloud staging area. When the command is complete a message appears stating that the cloud staging area successfully edited.

XML File Example

The following example shows editing an existing cloud staging area initially created with predefined target groups and target group mappings. In this scenario you can only modify **source_target_pairs**.

```
<?xml version='1.0' encoding='UTF-8'?>
<dmEditCloudStaging xmlns="http://schemas.teradata.com/dataMover/v2009"
xsi:schemaLocation="http://schemas.teradata.com/unity/DataMover.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <name>stagingarea</name>
  <storage_type>S3</storage_type>
  <source_target_pairs>
    <source_target_pair>
      <source_system>sourceSystem2</source_system>
    <source_system_target_group>sourceSystem2_stagingarea_tg</
source_system_target_group>
    <target_system>targetSystem2/target_system>
    <target_system_target_group>targetSystem2_stagingarea_tg</
target_system_target_group>
    </source_target_pair>
  </source_target_pairs>
</dmEditCloudStaging>
```

The following example shows editing an existing cloud staging area where Data Mover configured DSC for AWS S3, target groups and target group mappings. In this scenario you can modify **access_key_id**, **secret_access_key**, and **source_target_pairs**.

```
<?xml version='1.0' encoding='UTF-8'?>
<dmEditCloudStaging xmlns="http://schemas.teradata.com/dataMover/v2009"
xsi:schemaLocation="http://schemas.teradata.com/unity/DataMover.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <name>csa1</name>
```

```

<storage_type>S3</storage_type>
<s3_properties>
  <access_key_id>ABCDEFGH</access_key_id>
  <secret_access_key>AbcDEfgHIjklmNop123/456qRstUVwXyZ</secret_access_key>
  <buckets_by_regions>
    <buckets_by_region>
      <region>us-west-2</region>
      <buckets>
        <bucket>
          <bucket_name>my-s3-bucket</bucket_name>
          <prefix_list>
            <prefix>
              <prefix_name>backup</prefix_name>
              <storage_devices>100</storage_devices>
            </prefix>
          </prefix_list>
        </bucket>
      </buckets>
    </buckets_by_region>
  </buckets_by_regions>
</s3_properties>
<source_target_pairs>
  <source_target_pair>
    <source_system>sourceSystem1</source_system>
    <target_system>targetSystem1</target_system>
  </source_target_pair>
</source_target_pairs>
</dmEditCloudStaging>

```

encrypt_password

Purpose

The `encrypt_password` command creates an encrypted password from a password entered as a parameter on the command line, interactively, or through an XML file with a specific format. The generated password can then be used as a value for **-source_password_encrypted** or **-target_password_encrypted** when using the create and move commands.

Syntax

See [Data Mover XML Schemas](#).

Parameters

See [Parameter Order](#).

dir

[Optional] Name of the directory in which Data Mover stores the encrypted password file. If you specify a directory, but not a filename, the command returns the encrypted password to standard output.

Example:

```
/home/.dmauth
```

dm.rest.endpoint

[Optional] Enter a Data Mover REST server URL to overwrite the default value specified in the `commandline.properties` file in order to connect to a different REST server (and therefore a different daemon) at runtime.

```
https://dm-server1:1443/datamover
```

filename

[Optional] Name of an XML file to which Data Mover writes the encrypted password.

Example:

```
apw.xml
```

password

Password to encrypt. If you do not enter a password on the command line, the command prompts you to enter a password interactively. When you enter a password interactively, your input is masked with a set number of asterisks, regardless of the length of the password.

Example:

```
73r4| )474
```

XML File Examples

To use an XML file with the `encrypt_password` command to generate an encrypted password, type `datamove encrypt_password -f pw_parameters.xml`.

For security purposes, specifying passwords as plaintext and storing them in a file is not recommended. You can create an encrypted password file such as `pw_parameters.xml`, without saving a plaintext password in the file. In the following example, no value is provided for the `<password>` tag.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<dmGetEncryptedPassword xmlns="http://schemas.teradata.com/dataMover/v2009"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://schemas.teradata.com/unity/datamover.xsd">
  <password></password>
  <dir>/home/.dmauth</dir>
```



```
<filename>apw.xml</filename>
</dmGetEncryptedPassword>
```

Data Mover prompts you to enter the password, then writes the encrypted password to `/home/.dmath/apw.xml`.

```
/home/.dmath/apw.xml
-----
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<dmGetEncryptedPasswordOutput xmlns="http://schemas.teradata.com/dataMover/v2009"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  dmVersion="13.10.00.00"
  xsi:schemaLocation="http://schemas.teradata.com/unity/datamover.xsd">

  <password_encrypted>6683562fc7324fa8e49573c97ea9d4f76b6333f0749caa3a207c01f492231
  c8b</password_encrypted>
</dmGetEncryptedPasswordOutput>
```

The following example of a `pw_parameters.xml` file shows values for all tags.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<dmGetEncryptedPassword xmlns="http://schemas.teradata.com/dataMover/v2009"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://schemas.teradata.com/unity/datamover.xsd">
  <password>73r4|)474</password>
  <dir>/home/.dmath</dir>
  <filename>apw.xml</filename>
</dmGetEncryptedPassword>
```

Data Mover writes the encrypted password to `/home/.dmath/apw.xml`.

get_cloud_staging

Purpose

The `get_cloud_staging` allows you to retrieve the details of a cloud staging area.

Syntax

[Data Mover XML Schemas](#)

Parameters

See [Parameter Order](#).

name

Name of the cloud staging area to get details for.

Example:

my_staging_area

filename

[Optional] Name of the file where the details of the cloud staging area remains. If you do not provided the output file name, the command creates a file named with the cloud staging area name and .xml extension to store the details.

Example:

my_staging_area.xml

dm.rest.endpoint

[Optional] Enter a Data Mover REST server URL to overwrite the default value specified in the commandline.properties file in order to connect to a different REST server (and therefore a different daemon) at runtime.

https://dm-server1:1443/datamover

response_timeout

[Optional] Amount of time, in seconds, to wait for response from the Data Mover daemon.

Example:

60

security_password

[Optional] Password for the super user or authorized Viewpoint user.

Example:

53cUr17y

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password_encrypted** is also specified.

security_password_encrypted

[Optional] Encrypted password for the super user.

Example:

052c7aabd14c7770141ac3c0137ab98ae0d3f0f7cddf588981206b010c0c1b2f

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password** is also specified.

security_username

[Optional] User ID of the super user or authorized Viewpoint user. The user ID of the super user is dmc1_admin and cannot be changed.

Required if security management is enabled on the Data Mover daemon.

Usage Notes

You can use `get_cloud_staging` command with the command line parameter or an XML file. Type `datamove get_cloud_staging -name my_staging_area -filename output.xml` to get the details of a cloud staging area using the command line parameter. Type `datamove get_cloud_staging -f getcloudstaging.xml` to get the details of a cloud staging area using an XML file.

XML File Example

```
<?xml version='1.0' encoding='UTF-8'?>
<dmGetCloudStagingArea xmlns="http://schemas.teradata.com/dataMover/v2009"
xsi:schemaLocation="http://schemas.teradata.com/unity/DataMover.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <name>my_staging_area</name>
  <filename>my_output_file.xml</filename>
</dmGetCloudStagingArea>
```

help

Purpose

The `help` command gives a brief description of all of the commands. You can explore details about any Data Mover command by entering the command name after `help`. For example, to view parameter information and sample values for the `create` command, enter `datamove help create`.

Data Mover also displays help on the command line when you enter `--help` as a *parameter* to the `datamove` command. For example, the following two invocations of `help` on the command line are equivalent:

```
datamove help create
```

```
datamove --help create
```

list_agents

Purpose

The `list_agents` command lists the Data Mover agents connected to the Data Mover daemon.

Syntax

See [Data Mover XML Schemas](#).

Parameters

See [Parameter Order](#).

agent_names

[Optional] Lists the connectivity of the agents that are specified. When specifying the agents, use commas to separate the agents. Do not use spaces.

Example:

AgentA,AgentB

dm.rest.endpoint

[Optional] Enter a Data Mover REST server URL to overwrite the default value specified in the `commandline.properties` file in order to connect to a different REST server (and therefore a different daemon) at runtime.

`https://dm-server1:1443/datamover`

frequency

[Optional] Frequency for updating the status of the Data Mover agent in seconds.

Example:

5

If this parameter is specified, the status of the Data Mover agent is updated every frequency seconds. To stop the status updates, press the system INTR character (UNIX) or Ctrl+C (Windows). If the frequency parameter is not specified, the status is displayed only once and is not updated.

Usage Notes

If the `agent_names` parameter is not specified, the `list_agents` command lists all connected agents. If `agent_names` is used, only the connectivity of agents that are specified are listed.

XML File Example

For the `list_agents` command, type `datamove list_agents -f parameters.xml`.

The following example shows a parameters file for the `list_agents` command.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<dmListAgents
  xmlns="http://schemas.teradata.com/dataMover/v2009"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://schemas.teradata.com/unity/DataMover.xsd">
```

```
<agent_names>AgentA</agent_names>
</dmListAgents>
```

list_cloud_staging

Purpose

The `list_cloud_staging` allows you to get a list of existing cloud staging areas.

Syntax

[Data Mover XML Schemas](#)

Parameters

See [Parameter Order](#).

dm.rest.endpoint

[Optional] Enter a Data Mover REST server URL to overwrite the default value specified in the `commandline.properties` file in order to connect to a different REST server (and therefore a different daemon) at runtime.

`https://dm-server1:1443/datamover`

response_timeout

[Optional] Amount of time, in seconds, to wait for response from the Data Mover daemon.

Example:

`60`

security_password

[Optional] Password for the super user or authorized Viewpoint user.

Example:

`53cUr17y`

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password_encrypted** is also specified.

security_password_encrypted

[Optional] Encrypted password for the super user.

Example:

`052c7aabd14c7770141ac3c0137ab98ae0d3f0f7cddf588981206b010c0c1b2f`

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password** is also specified.

security_username

[Optional] User ID of the super user or authorized Viewpoint user. The user ID of the super user is `dmcl_admin` and cannot be changed.

Required if security management is enabled on the Data Mover daemon.

Usage Notes

You can use `list_cloud_staging` command with the command line parameter or an XML file. Type `datamove list_cloud_staging` to get a list of existing cloud staging areas. It is possible to use `datamove list_cloud_staging -f listcloudstaging.xml` and get the details of a cloud staging area using an XML file. but at this time there are no parameters you can provide in the XML file.

XML File Example

```
<?xml version='1.0' encoding='UTF-8'?>
<dmListCloudStagingAreas xmlns="http://schemas.teradata.com/dataMover/
v2009" xsi:schemaLocation="http://schemas.teradata.com/unity/DataMover.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
</dmListCloudStagingAreas>
```

list_configuration

Purpose

The `list_configuration` command outputs one file with configuration and performance settings for the daemon. Configuration settings for the event table are also included.

Syntax

[Data Mover XML Schemas](#)

Parameters

See [Parameter Order](#).

dir

[Optional] Output directory for the daemon configuration settings. A directory will be created if one does not exist.

Example:

`/home/datamover/config`

The generated XML file is written by default to the current directory.

dm.rest.endpoint

[Optional] Enter a Data Mover REST server URL to overwrite the default value specified in the `commandline.properties` file in order to connect to a different REST server (and therefore a different daemon) at runtime.

`https://dm-server1:1443/datamover`

filename

[Optional] Name with which to save the configuration file. Existing file name will be overwritten.

Example:

`configuration.xml`

The generated XML file has by default the name `configuration.xml`.

security_password

[Optional] Password for the super user or authorized Viewpoint user.

Example:

`53cUr17y`

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password_encrypted** is also specified.

security_password_encrypted

[Optional] Encrypted password for the super user.

Example:

`052c7aabd14c7770141ac3c0137ab98ae0d3f0f7cddf588981206b010c0c1b2f`

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password** is also specified.

security_username

[Optional] User ID of the super user or authorized Viewpoint user. The user ID of the super user is `dmc1_admin` and cannot be changed.

Required if security management is enabled on the Data Mover daemon.

Usage Notes

Edit the file information, then save the file to the daemon with the `save_configuration` command.

Configurable options include settings for:

- Data dictionary views
- Forcing the direction of copying between source and target Teradata Database systems

- Online archiving
- Preventing a Teradata Database system from being used as a target
- Overwriting existing objects
- Messages that are generated for the event table
- Performance tuning
- Interaction with Teradata Ecosystem Manager
- Management of security for the command-line interface and the Data Mover portlet
- Client encryption

XML File Example

For the `list_configuration` command, type `datamove list_configuration -f parameters.xml`.

In the following example, the parameters file writes the Data Mover daemon configuration to `/home/datamover/config/configuration.xml`.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<dmListConfiguration
  xmlns="http://schemas.teradata.com/dataMover/v2009"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://schemas.teradata.com/unity/DataMover.xsd">
  <dir>/home/datamover/config</dir>
  <filename>configuration.xml</filename>
</dmListConfiguration>
```

list_event_table

Purpose

The `list_event_table` command displays a list of event tables.

Syntax

[Data Mover XML Schemas](#)

Parameters

See [Parameter Order](#).

dm.rest.endpoint

[Optional] Enter a Data Mover REST server URL to overwrite the default value specified in the `commandline.properties` file in order to connect to a different REST server (and therefore a different daemon) at runtime.

`https://dm-server1:1443/datamover`

security_password

[Optional] Password for the super user or authorized Viewpoint user.

Example:

53cUr17y

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password_encrypted** is also specified.

security_password_encrypted

[Optional] Encrypted password for the super user.

Example:

052c7aabd14c7770141ac3c0137ab98ae0d3f0f7cddf588981206b010c0c1b2f

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password** is also specified.

security_username

[Optional] User ID of the super user or authorized Viewpoint user. The user ID of the super user is `dmc1_admin` and cannot be changed.

Required if security management is enabled on the Data Mover daemon.

list_job_definition

Purpose

The `list_job_definition` command lists the job definition for the job name you specify.

You can modify job behavior either of the following two methods:

- Edit the object list and job parameters in the job definition. Then use the `create` command to save the job modifications to the daemon, using a new name to create a new job definition.
- Supply new job values for job variables at runtime using the `start` command.

If you specify the original job name for the `list_job_definition` command, the original job definition is listed. If you specify the name of a job instance in which job variable values were modified at runtime, the modified job definition is listed.

Syntax

[Data Mover XML Schemas](#)

Parameters

See [Parameter Order](#).

dm.rest.endpoint

[Optional] Enter a Data Mover REST server URL to overwrite the default value specified in the `commandline.properties` file in order to connect to a different REST server (and therefore a different daemon) at runtime.

`https://dm-server1:1443/datamover`

dir

[Optional] Output directory for the job definition. A directory will be created if one does not exist.

Example:

`/home/datamover/jobdefs`

The generated XML file is written by default to the current directory.

filename

[Optional] Output file name for the job definition. Existing file name will be overwritten.

Example:

`jobdefinition.xml`

The generated XML file has by default the name `<job_name>.xml`.

job_name

Name of the job definition that will be output.

Example:

`12315DFHJKS`

security_password

[Optional] Password for the super user or authorized Viewpoint user.

Example:

`53cUr17y`

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password_encrypted** is also specified.

security_password_encrypted

[Optional] Encrypted password for the super user.

Example:

`052c7aabd14c7770141ac3c0137ab98ae0d3f0f7cddf588981206b010c0c1b2f`

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password** is also specified.

security_username

[Optional] User ID of the super user or authorized Viewpoint user. The user ID of the super user is `dmc1_admin` and cannot be changed.

Required if security management is enabled on the Data Mover daemon.

XML File Example

For the `list_job_definition` command, type `datamove list_job_definition -f parameters.xml`.

In the following example, the parameters file writes the definition for the 12315DFHJKS to `/home/datamover/jobdefs/12315DFHJKS_jobdefinition1.xml`.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<dmGetJobDefinition
xmlns="http://schemas.teradata.com/dataMover/v2009"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schemas.teradata.com/unity/DataMover.xsd">
  <job_name>12315DFHJKS</job_name>
  <dir>/home/datamover/jobdefs</dir>
  <filename>12315DFHJKS_jobdefinition1.xml</filename>
</dmGetJobDefinition>
```

Here is an example of an output file produced when a user runs the `list_job_definition` command:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<dmCreate xmlns="http://schemas.teradata.com/dataMover/v2009"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://
schemas.teradata.com/unity/DataMover.xsd">
  <job_name>jp_low_5</job_name>
  <job_priority>LOW</job_priority>
  dmdev
  <source_user></source_user>
  <source_password></source_password>
  dmsmp
  <target_user></target_user>
  <target_password></target_password>
  <use_userid_pool>>false</use_userid_pool>
  <data_streams>1</data_streams>
  <source_sessions>1</source_sessions>
  <target_sessions>1</target_sessions>
  <max_agents_per_task>1</max_agents_per_task>
```

```

<overwrite_existing_objects>TRUE</overwrite_existing_objects>
<freeze_job_steps>TRUE</freeze_job_steps>
<force_utility>JDBC</force_utility>
<compare_ddl>UNSPECIFIED</compare_ddl>
<log_level>1</log_level>
<online_archive>UNSPECIFIED</online_archive>
<database selection="unselected">
  <name>DB</name>
  <table selection="included">
    <name>TEST101</name>
    <compare_ddl>FALSE</compare_ddl>
  </table>
</database>
<dmCreate>

```

list_job_steps

Purpose

The `list_job_steps` command displays all of the steps in a job plan. Use this information in a start command to control which steps to start.

If you specify the original job name for the `list_job_steps` command, the original job steps listed may be outdated. Specify the job execution name to view the exact job steps.

Syntax

[Data Mover XML Schemas](#)

Parameters

See [Parameter Order](#).

dir

[Optional] Output directory for the list of job steps. A directory will be created if one does not already exist.

`/home/datamover/steps`

The generated XML file is written by default to the current directory.

dm.rest.endpoint

[Optional] Enter a Data Mover REST server URL to overwrite the default value specified in the `commandline.properties` file in order to connect to a different REST server (and therefore a different daemon) at runtime.

`https://dm-server1:1443/datamover`

filename

[Optional] Output file name for the list of job steps. If the file already exists, it will be overwritten.
The generated XML file has by default the name `<job_name>_jobSteps.xml`.

job_name

Name of the job.

Example:

12333KZIHRT

security_password

[Optional] Password for the super user or authorized Viewpoint user.

Example:

53cUr17y

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password_encrypted** is also specified.

security_password_encrypted

[Optional] Encrypted password for the super user.

Example:

052c7aabd14c7770141ac3c0137ab98ae0d3f0f7cddf588981206b010c0c1b2f

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password** is also specified.

security_username

[Optional] User ID of the super user or authorized Viewpoint user. The user ID of the super user is `dmcl_admin` and cannot be changed.

Required if security management is enabled on the Data Mover daemon.

XML File Example

For the `list_job_steps` command, type `datamove list_job_steps -f parameters.xml`.

In the following example, the parameters file writes job steps for 12333KZIHRT to `/home/datamover/steps/12333KZIHRT_jobSteps.xml`.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<dmListJobSteps>
  <job_name>12333KZIHRT</job_name>
  <dir>/home/datamover/steps</dir>
```

```
<filename>12333KZIHRT_jobSteps.xml</filename>
</dmListJobSteps>
```

list_jobs

Purpose

The `list_jobs` command displays a list of all jobs that have run on the Data Mover daemon with the specified status mode. Job definitions are not displayed.

Syntax

[Data Mover XML Schemas](#)

Parameters

See [Parameter Order](#).

dm.rest.endpoint

[Optional] Enter a Data Mover REST server URL to overwrite the default value specified in the `commandline.properties` file in order to connect to a different REST server (and therefore a different daemon) at runtime.

`https://dm-server1:1443/datamover`

end_time

[Optional] Lists all jobs that finished earlier than the specified time.

The format is: `yyyy-MM-dd-HH` in local time.

Example:

`2019-02-20-16`

Note:

`end_time` cannot be used with `last_hour`.

freeze_step_only

[Optional] Lists only jobs with `freezeJobSteps=true`.

last_hour

[Optional] Lists all jobs that finished in the last X hours, where X is the number of hours.

The value must be a positive integer.

Example:

`6`

Note:

`last_hour` cannot be used with `start_time` or `end_time`.

latest_job_only

[Optional] Returns the status of the most recent completed job, per job name, after meeting the conditions specified by other parameters.

For example, if used with the parameter **status_mode** set to failed (F), only the latest completed job instances with a failed status are returned.

If the **end_time** parameter is used, the latest job instances that completed before the specified end time are returned.

start_time

[Optional] Lists all jobs that started later than the specified time.

The format is: yyyy-MM-dd-HH in local time.

Example:

2019-02-20-16

Note:

`start_time` cannot be used with `last_hour`.

status_mode

[Optional] Lists all jobs with the status mode specified in the following table.

Example:

R

job_name

[Optional] Name of the job to filter the results with.

Example:

12315DFHJKS

Status Mode	Description
A	All jobs (default).
N	All new jobs.
I	All initializing jobs.

Status Mode	Description
R	All running jobs.
C	All successfully completed and completed with warnings jobs.
F	All failed jobs.
RS	All restarting jobs.
Q	All queued jobs.
B	All blocked jobs due to object locks.
UC	All user-cancelled jobs.

XML File Example

For the `list_jobs` command, type `datamove list_jobs -f parameters.xml`.

The following example shows a parameters file, `dmListJobs.xml`, for the `list_jobs` command.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<dmListJobs xmlns="http://schemas.teradata.com/dataMover/v2009"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schemas.teradata.com/unity/DataMover.xsd">
    <status_mode>C</status_mode>
</dmListJobs>
```

Here is an example of the screen output of running the `list_jobs` command:

```
Data Mover Command Line 16.20.25.00
2019-05-13 18:56:22 - List Jobs mode
Command parameters:
About to connect to REST server at https://localhost:1443/datamover
Connected to Daemon version 16.20.25.00
List jobs...
Job Name                                     Start
End                                     Status                                     Priority
-----
jp_high_1      5/13/19 6:50 PM      5/13/19 6:50 PM      COMPLETED_SUCCESSFULLY HIGH
jp_high_2      5/13/19 6:51 PM      5/13/19 6:51 PM      COMPLETED_SUCCESSFULLY HIGH
jp_high_3      5/13/19 6:51 PM      5/13/19 6:52 PM      COMPLETED_SUCCESSFULLY HIGH
jp_high_4      5/13/19 6:52 PM      5/13/19 6:52 PM      COMPLETED_SUCCESSFULLY HIGH
jp_low_1       5/13/19 6:53 PM      5/13/19 6:53 PM      COMPLETED_SUCCESSFULLY LOW
jp_low_2       5/13/19 6:54 PM      5/13/19 6:54 PM      COMPLETED_SUCCESSFULLY LOW
jp_low_3       5/13/19 6:54 PM      5/13/19 6:55 PM      COMPLETED_SUCCESSFULLY LOW
jp_low_4       5/13/19 6:55 PM      5/13/19 6:55 PM      COMPLETED_SUCCESSFULLY LOW
```


list_tasks

Purpose

Tasks are utility-specific and are executed by the Data Mover agent. From a high-level, perspective, this view identifies tasks using Data Mover's task-related resources, such as agent task slots, load slots, and so forth. From a job-level perspective, it identifies work that is blocking a particular job and how long of a delay before execution.

Syntax

[Data Mover XML Schemas](#)

Parameters

See [Parameter Order](#).

agent_name

[Optional] Agent name.

AgentA

dm.rest.endpoint

[Optional] Enter a Data Mover REST server URL to overwrite the default value specified in the `commandline.properties` file in order to connect to a different REST server (and therefore a different daemon) at runtime.

`https://dm-server1:1443/datamover`

frequency

[Optional] Frequency of status update in seconds.

Example:

5

job_name

Example:

[Optional] Monitors tasks related to this job.

12315DFHJKS

security_password

[Optional] Password for the super user or authorized Viewpoint user.

Example:

53cUr17y

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password_encrypted** is also specified.

security_password_encrypted

[Optional] Encrypted password for the super user.

Example:

052c7aabd14c7770141ac3c0137ab98ae0d3f0f7cddf588981206b010c0c1b2f

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password** is also specified.

security_username

[Optional] User ID of the super user or authorized Viewpoint user. The user ID of the super user is `dmc1_admin` and cannot be changed.

Required if security management is enabled on the Data Mover daemon.

task_id

[Optional] Task ID.

Example:

35

task_status_mode

[Optional] Lists all tasks with the status mode specified.

Example:

R

Status Mode	Description
A	All active tasks (default)
R	All tasks currently running
Q	All tasks currently queued

Usage Notes

- To stop the task view updates if you supplied a frequency parameter, press the system INTR character (UNIX) or Ctrl+C (Windows).
- The order of XML elements is significant and must be observed.

XML File Example

For the `list_tasks` command, type `datamove list_tasks -f parameters.xml`.

The following example shows all queued tasks for job `job1` and taskid `123` on agent `AgentA`, and updates the view every five seconds.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<dmListTasks xmlns="http://schemas.teradata.com/dataMover/
v2009" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schemas.teradata.com/unity/DataMover.xsd">
<task_status_mode>Q</task_status_mode>
<job_name>job1</job_name>
<task_id>123</task_id>
<agent_name>AgentA</agent_name>
<frequency>5</frequency>
</dmListTasks>
```

Because all parameters are option for the `list_tasks` command, the following example shows a minimal XML file.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<dmListTasks xmlns="http://schemas.teradata.com/dataMover/v2009"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://
schemas.teradata.com/unity/DataMover.xsd">
</dmListTasks>
```

Command Output Example

The following example is an excerpt of the output from the `list_tasks` command.

```
Data Mover Command Line 15.10.00.00
```

```
9/10/14 4:32:12 PM
```

```
list_tasks mode
```

```
Command parameters:
```

```
- status_mode: ALL
```

```
About to connect to ActiveMQ at localhost:61616
```

```
Requesting status...
```

```
Connected to Daemon version 15.10.00.00
```

TASK	JOB NAME	PRIORITY	STATUS	AGENT	QUEUE	QUEUE ORDER	UTILITY	SRC SYSTEM	TARGET SYSTEM	USER POOL
133	arc3	HIGH	QUEUED	-	USER_POOL	-	ARC	pmdev1	calvin	pool1
243	arc34	MEDIUM	QUEUED	-	USER_POOL	-	ARC	pmdev1	calvin	reservedU
123	hdconn	LOW	QUEUED	-	HD_CONNECTOR_SLOT 1	1	HADOOP_CONNECTOR	http://t3:60070	gmdev	-
13	hdcon2	LOW	QUEUED	-	HD_CONNECTOR_SLOT 2	2	HADOOP_CONNECTOR	http://t4:60070	gmdev	-
176	tpt	MEDIUM	QUEUED	-	LOAD_SLOT	1	TPTAPI_LOAD	pmdev1	gmprod	-
126	tpt1	MEDIUM	QUEUED	-	LOAD_SLOT	2	TPTAPI_LOAD	pmdev1	gmprod	-
136	tpt3	MEDIUM	QUEUED	-	LOAD_SLOT	3	TPTAPI_UPDATE	pmdev1	gmprod	-
62	hdsq1h	HIGH	QUEUED	-	SQLH_SLOT	1	SQLH	http://t1:50070	http://t2:50070	-
134	jdbc	HIGH	QUEUED	-	AGENT_SLOT	1	JDBC	smdev1	calvin	-
125	arcgp	HIGH	RUNNING	agent1	NOT_QUEUED	-	ARC	smdev1	calvin	pool1
125	arcgp	HIGH	RUNNING	agent6	NOT_QUEUED	-	ARC	smdev1	calvin	pool1
155	arcgp2	HIGH	RUNNING	agent3	NOT_QUEUED	-	ARC	smdev1	calvin	pool12
155	arcgp2	HIGH	RUNNING	agent6	NOT_QUEUED	-	ARC	smdev1	calvin	pool12

Output columns include the following as required or applicable:

- TASK
- JOB NAME
- PRIORITY
- STATUS
- AGENT
- QUEUE
- QUEUE ORDER
- UTILITY
- SRC SYSTEM
- TARGET SYSTEM
- USER POOL
- LAST UPDATE

Security

- The `list_tasks` command follows the same security checks as the `list_jobs` command.
- When security is not enabled, a command-line user has permission to run all Data Mover commands, which includes the `list_tasks` command.
- When security is enabled, a user needs job-level permission (if security is set to job level) and daemon-level read permission to run the `list_tasks` command.

Error Messages and Other Output

The following error messages cause the Data Mover command-line interface to exit with a failure code of -1:

Message	Description
- task_id: Error: [value] is not numeric.	task_id is specified but cannot be converted to a numeric number.
- task_status_mode: Error: Invalid task_status_mode: [value]. See list_tasks help for the list of valid options.	task_status_mode is specified but with an incorrect value.
- frequency: [value] is not an integer, value specified has an invalid format or the value is greater than 2147483647.	A frequency is specified but cannot be converted to an integer.
- frequency: [value] is not a positive integer.	A frequency is specified but not as a positive integer.
Error: Active tasks found but not visible due to security restriction. User must have read permission on job to view job's tasks.	Security is enabled and the user supplied does not have permission to view the tasks.

modify_admin_password

Purpose

The modify_admin_password command allows you to change the password for the super user. By default, the password for the super user is dmcl_admin and cannot be changed.

Syntax

[Data Mover XML Schemas](#)

Parameters

See [Parameter Order](#).

current_password

Current password for the super user. By default, password of the super user is dmcl_admin.

Example:

dmcl_admin

Not a valid parameter if **-current_password_encrypted** is also specified.

current_password_encrypted

Current encrypted password for the super user.

Example:

8fe40e30c56c8f3217b02f978ee347997b5ad82b3f8ac532384484a76cf25be3

Not a valid parameter if **-current_password** is also specified.

dm.rest.endpoint

[Optional] Enter a Data Mover REST server URL to overwrite the default value specified in the `commandline.properties` file in order to connect to a different REST server (and therefore a different daemon) at runtime.

`https://dm-server1:1443/datamover`

new_password

New password for the super user.

Example:

`53cUr17y`

Not a valid parameter if **-new_password_encrypted** is also specified.

new_password_encrypted

New encrypted password for the super user.

Example:

`052c7aabd14c7770141ac3c0137ab98ae0d3f0f7cddf588981206b010c0c1b2f`

Not a valid parameter if **-new_password** is also specified.

security_password

[Optional] Password for the super user or authorized Viewpoint user.

Example:

`53cUr17y`

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password_encrypted** is also specified.

security_password_encrypted

[Optional] Encrypted password for the super user.

Example:

`052c7aabd14c7770141ac3c0137ab98ae0d3f0f7cddf588981206b010c0c1b2f`

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password** is also specified.

security_username

[Optional] User ID of the super user or authorized Viewpoint user. The user ID of the super user is `dmc1_admin` and cannot be changed.

Required if security management is enabled on the Data Mover daemon.

Usage Notes

You can pass values for the following:

- A current plaintext password and a new encrypted password in the same command or XML file.
- A current encrypted password and a new plaintext password in the same command or XML file.

The following scenarios result in an error:

- You pass values for the current plaintext and current encrypted passwords in the same command or XML file.
- You pass values for the new plaintext and new encrypted passwords in the same command or XML file.

XML File Example

To use an XML file with the `modify_admin_password` command to change the password for the super user, type `datamove modify_admin_password -f admin_pw_parameters.xml`.

In the following example, the current (default) password is updated to 53cUr17y.

```
<dmModifyAdminPassword xmlns="http://schemas.teradata.com/dataMover/v2009"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://schemas.teradata.com/unity/DataMover.xsd">
  <current_password>dmcl_admin</current_password>
  <current_password_encrypted></current_password_encrypted>
  <new_password>53cUr17y</new_password>
  <new_password_encrypted></new_password_encrypted>
</dmModifyAdminPassword>
```

To pass values for encrypted passwords, add them to the XML file for the appropriate element.

modify_event_table

Purpose

The `modify_event_table` command enables you to change the Teradata user name and password for an existing Event table.

Syntax

[Data Mover XML Schemas](#)

Parameters

See [Parameter Order](#).

change_user_name

Determines whether or not the user name is changed.

Example:

true

change_user_password

Determines whether or not the user password is changed.

dm.rest.endpoint

[Optional] Enter a Data Mover REST server URL to overwrite the default value specified in the `commandline.properties` file in order to connect to a different REST server (and therefore a different daemon) at runtime.

`https://dm-server1:1443/datamover`

event_table_name

Label that uniquely identifies the installation of the Event table.

Example:

`event1`

security_password

[Optional] Password for the super user or authorized Viewpoint user.

Example:

`53cUr17y`

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password_encrypted** is also specified.

security_password_encrypted

[Optional] Encrypted password for the super user.

Example:

`052c7aabd14c7770141ac3c0137ab98ae0d3f0f7cddf588981206b010c0c1b2f`

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password** is also specified.

security_username

[Optional] User ID of the super user or authorized Viewpoint user. The user ID of the super user is `dmc1_admin` and cannot be changed.

Required if security management is enabled on the Data Mover daemon.

user_name

Existing Teradata user name for Data Mover to use to access the Event table. The user requires INSERT permission for the TMSMEvent table..

Example:

dmuser1

user_password

Password for the Teradata user associated with the user_name.

Example:

dmuser1pass

XML File Example

For the create_event_table command, type `datamove create_event_table -f parameters.xml`.

The following example is a parameters file for the modify_event_table command.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<dmModifyEventTable xmlns="http://schemas.teradata.com/dataMover/v2009"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schemas.teradata.com/unity/DataMover.xsd">
  <event_table_name>eventTable1</event_table_name>
  <change_user_name>true</change_user_name>
  <user_name>tduser2</user_name>
  <change_user_password>true</change_user_password>
  <user_password>tdpass2</user_password>
</dmModifyEventTable>
```

move

The move command creates a job to copy specified database objects from one database to another, then starts the job immediately. Use the command to create and start a job quickly without the XML object list by specifying tables on the command line.

Available parameters for this command may vary depending on the involved systems, the direction of data movement, and whether you are using the CLI or an .xml file. Refer to the parameter descriptions for guidance on the context in which you can use the parameters outlined in a section.

move

Purpose

The move command creates a job to copy specified database objects from one database to another, then starts the job immediately. Use the command to create and start a job quickly without the XML object list by specifying tables on the command line.

Note:

For jobs that use Teradata DSA, the database user for the target system to which you are copying tables must have RESTORE privileges and must not be logged on while the job is running or the job will fail.

Syntax

[Data Mover XML Schemas](#)

Parameters

See [Parameter Order](#).

data_streams

[Optional] Number of data streams to use between the source and target databases. Applies to jobs that use Teradata DSA and TPT API (to and from Teradata). All other protocols use a single data stream.

Example:

4

The default value is dynamically calculated by Data Mover.

db_client_encryption

[Optional] Set to true if job needs to be encrypted during data transfer.

dm.rest.endpoint

[Optional] Enter a Data Mover REST server URL to overwrite the default value specified in the `commandline.properties` file in order to connect to a different REST server (and therefore a different daemon) at runtime.

`https://dm-server1:1443/datamover`

force_utility

[Optional] Forces the Data Mover daemon to use a specific utility for all copy operations.

Valid Values

- dsa

- jdbc
- tptapi
- tptapi_load
- tptapi_stream
- tptapi_update
- T2T

If this value is not specified, the Data Mover daemon determines which Teradata utility is the best to use for the job.

Note:

Copying data to an older version of the Teradata Database using Teradata DSA is not valid. You cannot use Teradata DSA if the source and target TDPIDs are the same.

Example:

dsa

job_name

[Optional] Name for this job. The name must be unique and up to 32 characters. If the name is not specified, it is automatically generated in the form *<source_tdpid>_<target_tdpid>_M<sequential number>*.

job_priority

[Optional] Specifies the execution priority for the job. Supported values are: HIGH, MEDIUM, and LOW, and UNSPECIFIED. If no value is specified, the default of MEDIUM is used at runtime.

Example:

MEDIUM

log_level

[Optional] Log level for log file output.

Valid Values

- 0
- 1
- 2
- 99

Example:

2

The default value is 0.

max_agents_per_task

[Optional] Maximum number of Data Mover agents to use in parallel when moving tables or databases.

Example:

4

The default value is dynamically calculated by Data Mover.

netrace

[Optional] CLI netrace parameter. Any value greater than or equal to 0 generates a CLI trace log. Valid CLI value must be provided.

netrace_buf_len

[Optional] CLI netrace_buf_len parameter. Any value greater than or equal to 0 generates a CLI trace log. Valid CLI value must be provided.

online_archive

[Optional] Allows read and write access to the source table(s) while the tables are being copied with Teradata DSA. Updates occur to the source table during the copy, but are not transferred to the target table. After a successful copy, the data contained in the target table matches the data that was in the source table at the beginning of the copy.

Valid Values

Value	Description
True	Enables online archive
False	Disables online archive
Unspecified	Default – the value is set to the value in the Data Mover daemon configuration file

Example:

true

overwrite_existing_objects

[Optional] Job overwrites objects that already exist on the target.

Valid Values

Value	Description
True	Enables overwriting
False	Disables overwriting

Value	Description
Unspecified	Default – the value is set to the value in the Data Mover daemon configuration file

If the parameter is not specified, the value is set to the `overwrite_existing_objects` parameter value in the Data Mover daemon configuration file. If the parameter is specified as `true` or `false`, that value takes precedence over the parameter value in the Data Mover daemon configuration file.

Example:

```
true
```

query_band

[Optional] A semicolon-separated set of name-value pairs that uniquely identifies Teradata sessions or transactions for the source and target. To use a query band to identify the job payroll, the user ID aa100000, and job session number 1122, define the query band as follows:

Example:

```
Job=payroll;Userid=aa100000;Jobsession=1122;
```

This parameter must be added as the last parameter in the XML job definition. See [About Query Band](#) for detailed syntax rules when defining a query band.

response_timeout

[Optional] Amount of time, in seconds, to wait for response from the Data Mover daemon.

Example:

```
60
```

security_password

[Optional] Password for the super user or authorized Viewpoint user.

Example:

```
53cUr17y
```

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password_encrypted** is also specified.

security_username

[Optional] User ID of the super user or authorized Viewpoint user. The user ID of the super user is `dmc1_admin` and cannot be changed.

Required if security management is enabled on the Data Mover daemon.

source_account_id

[Optional] Logon account ID for source database.

Note:

Spaces in the account name for the source or target account id causes the job to fail.

source_logon_mechanism

[Optional] Logon mechanism for source system. To log on to a source Teradata Database system, the user must provide at least one of the following:

- `source_user` *and* `source_password`
- `source_logon_mechanism`

Logon mechanisms are not supported for Teradata DSA jobs. Use logon mechanisms only for Teradata PT API and Teradata JDBC jobs. If **-source_logon_mechanism** is specified and **-force_utility** is not used, Teradata PT API is used by default. Specifying **-source_logon_mechanism** with Teradata DSA specified for **-force_utility** results in an error.

Example:

KRB5

source_logon_mechanism_data

[Optional] Additional parameters needed for the logon mechanism of the source system.

Example:

joe@domain1@@mypassword

source_password

[Optional] Source Teradata logon password.

Example:

123456789

Not a valid parameter if **-source_password_encrypted** is also specified. If you do not specify a password for this parameter, the command prompts you to enter it interactively. Input is masked with a set number of asterisks, regardless of the length of the password.

source_password_encrypted

[Optional] Source Teradata encrypted logon password.

Example:

17894cc84b5637a88e36fa37a010e3662d18f64b8ce204bef8d63868ad417810

Not a valid parameter if **-source_password** is also specified.

source_sessions

[Optional] Number of sessions per data stream on the source database.

Example:

4

The default value is dynamically calculated by Data Mover.

source_tdpid

Source Teradata Database.

Example:

Checks

source_user

[Optional] Source Teradata logon id.

Example:

TD_API_user

If you do not specify a logon id for this parameter, the command prompts you to enter it interactively.

Note:

Spaces in the user name for the source or target id causes the job to fail.

source_userid_pool

[Optional] Job pulls the user from the specified credential pool. Available for any job type. Must use the same credential pool as `target_userid_pool` if specifying both parameters in the same job definition.

Example:

POOL-1

sync

[Optional] Waits for the job to complete, then returns an exit code that indicates if the job completed successfully. An exit code of 0 indicates successful job completion. An exit code other than 0 indicates an error with the job or with the command.

table

[Optional] Table to be copied.

Example:

DB1.TABLE

target_account_id

[Optional] Logon account ID for target database.

Note:

Spaces in the account name for the source or target account id causes the job to fail.

target_logon_mechanism

[Optional] Logon mechanism for target system. To log on to a target Teradata Database system, the user must provide at least one of the following:

- target_user *and* target_password
- target_logon_mechanism

Teradata DSA does not support logon mechanisms. Use logon mechanisms only with Teradata PT API and Teradata JDBC jobs. If **-target_logon_mechanism** is specified and **-force_utility** is not used, Teradata PT API is used by default. Specifying **-target_logon_mechanism** with Teradata DSA specified for **-force_utility** results in an error.

Example:

KRB5

target_logon_mechanism_data

[Optional] Additional parameters that are needed for the target system's logon mechanism.

Example:

my@domain2@@mypassword

target_password

[Optional] Target Teradata logon password.

Example:

212133344

Not a valid parameter if **-target_password_encrypted** is also specified. If you do not specify a password for this parameter, the command prompts you to enter it interactively. Input is masked with a set number of asterisks, regardless of the length of the password.

target_password_encrypted

[Optional] Target Teradata encrypted logon password.

Example:

```
30e458fce484cefef07724653f5046095208f69fcfbf76bf7290b8576192c2fe
```

Not a valid parameter if **-target_password** is also specified.

target_sessions

[Optional] Number of sessions per data stream on the target database.

Example:

```
4
```

The default value is dynamically calculated by Data Mover.

target_tdpid

[Optional] Target Teradata Database.

Example:

```
Leo
```

target_user

[Optional] Target Teradata logon id.

Example:

```
TD_tar_User
```

If you do not specify a logon id for this parameter, the command prompts you to enter it interactively.

Note:

Spaces in the user name for the source or target id causes the job to fail.

target_userid_pool

[Optional] Job pulls the user from the specified credential pool. Available for any job type. Must use the same credential pool as `source_userid_pool` if specifying both parameters in the same job definition.

Example:

```
POOL-1
```

tpt_debug

[Optional] TPT API trace debug log parameter. Any value greater than or equal to 0 generates a TPT API trace log. Valid TPT API value must be provided.

uowid

[Optional] Alternate ID or name for the batch of work associated with the job. If you provide a value for this parameter, Data Mover reports this value as the unit of work ID when sending events to Teradata Ecosystem Manager or to its internal TMSMEVENT table. If you do not specify this parameter, Data Mover uses a default value as the unit of work ID when sending events to Teradata Ecosystem Manager or to its internal TMSMEVENT table. The default value for the unit of work ID is composed of the job execution name and the current timestamp. For example, if you want to define the origins of a query source the job execution name is `sales_table`, the default value of the unit of work ID is `sales_table-20211110122330`

Example:

`sales_tables_move`

XML File Example

For the move command, type `datamove move -f parameters.xml`.

In the following example, the `<dmCreate>` element is used in the XML file, instead of the `<dmMove>` element.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<dmCreate xmlns="http://schemas.teradata.com/dataMover/v2009"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schemas.teradata.com/unity/DataMover.xsd">
  <job_name>floyd_dmdev_create</job_name>
  <source_tdpid>floyd</source_tdpid>
  <source_user>dmguest</source_user>
  <source_password>please</source_password>
  <target_tdpid>dmdev</target_tdpid>
  <target_user>dmguest</target_user>
  <target_password>please</target_password>
  <data_streams>5</data_streams>
  <source_sessions>1</source_sessions>
  <target_sessions>1</target_sessions>
  <force_utility>dsa</force_utility>
  <log_level>0</log_level>
  <database selection="unselected">
    <name>dmguest</name>
    <table selection="included">
      <name>test1</name>
    </table>
    <table selection="included">
      <name>test2</name>
    </table>
  </database>
</dmCreate>
```

```

    <table selection="included">
      <name>test3</name>
    </table>
  </database>
  <uowid>sales_tables_move</uowid>
  <query_band>Job=payroll;Userid=aa1000000;Jobsession=1122;</query_band>
</dmCreate>

```

restart

Purpose

The restart command restarts a job that has failed.

Syntax

[Data Mover XML Schemas](#)

Parameters

See [Parameter Order](#).

dm.rest.endpoint

[Optional] Enter a Data Mover REST server URL to overwrite the default value specified in the `commandline.properties` file in order to connect to a different REST server (and therefore a different daemon) at runtime.

`https://dm-server1:1443/datamover`

job_name

Name of the job to be restarted.

Example:

`12315DFHJKS`

security_password

[Optional] Password for the super user or authorized Viewpoint user.

Example:

`53cUr17y`

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password_encrypted** is also specified.

security_password_encrypted

[Optional] Encrypted password for the super user.

Example:

052c7aabd14c7770141ac3c0137ab98ae0d3f0f7cddf588981206b010c0c1b2f

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password** is also specified.

security_username

[Optional] User ID of the super user or authorized Viewpoint user. The user ID of the super user is `dmc1_admin` and cannot be changed.

Required if security management is enabled on the Data Mover daemon.

sync

[Optional] Waits for the job to complete, then returns an exit code that indicates if the job completed successfully. An exit code of 0 indicates successful job completion. An exit code other than 0 indicates an error with the job or with the command.

uowid

[Optional] Alternate ID or name for the batch of work associated with the job. If you provide a value for this parameter, Data Mover reports this value as the unit of work ID when sending events to Teradata Ecosystem Manager or to its internal TMSMEVENT table. If you do not specify this parameter, Data Mover uses a default value as the unit of work ID when sending events to Teradata Ecosystem Manager or to its internal TMSMEVENT table. The default value for the unit of work ID is composed of the job execution name and the current timestamp. For example, if you want to define the origins of a query source the job execution name is `sales_table`, the default value of the unit of work ID is `sales_table-20211110122330`.

Example:

`sales_tables_restart`

XML File Example

For the restart command, type `datamove restart -f parameters.xml`.

In the following example, the parameters file restarts the job 12315DFHJKS, and reports `sales_table_restart` to Teradata Ecosystem Manager as the unit of work ID.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<dmStart
  xmlns="http://schemas.teradata.com/dataMover/v2009"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://schemas.teradata.com/unity/DataMover.xsd">
  <job_name>12315DFHJKS</job_name>
```

```
<uowid>sales_tables_restart</uowid>
</dmStart>
```

restore_daemon

Purpose

The `restore_daemon` command allows you to restore the previously backed up daemon configuration from a repository in to a configuration file.

As of Data Mover 17.05, the `restore_daemon` command uses the `repository_backup.sh` scripts located in the `/opt/teradata/datamover/daemon/nn.nn` directory to restore daemon configurations from the repository. The script uses the Postgres `pg_restore` command for restoring.

Note:

The `repository_backup.sh` script can be used independently for restoring the daemon. Avoid restoring the daemon while jobs are running.

Note:

Where *nn.nn* in the path refers to the version numbers of Data Mover.

Syntax

[Data Mover XML Schemas](#)

Parameters

See [Parameter Order](#).

backup_target_dir

The directory from which Data Mover restores the backup files. The location you specify can be:

- A path relative to the `daemon_backup` directory. This is a directory reserved for all backup snapshots.
If the path is relative, enter the name of the directory that contains the backup files. Never use an initial slash when specifying a relative path.
- An absolute path. This is any location on the system for which `dm_user` has write privilege. `restore_daemon` runs under the `dm_user` account.
If the path is absolute, enter the full path to the backup files. Always use an initial slash when specifying an absolute path.

Example:

`repos_bu_001` is a relative path.

Example:

/home/myhome/repos_bu001 is an absolute path.

dm.rest.endpoint

[Optional] Enter a Data Mover REST server URL to overwrite the default value specified in the `commandline.properties` file in order to connect to a different REST server (and therefore a different daemon) at runtime.

`https://dm-server1:1443/datamover`

security_password

[Optional] Password for the super user or authorized Viewpoint user.

Example:

`53cUr17y`

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password_encrypted** is also specified.

security_password_encrypted

[Optional] Encrypted password for the super user.

Example:

`052c7aabd14c7770141ac3c0137ab98ae0d3f0f7cddf588981206b010c0c1b2f`

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password** is also specified.

security_username

[Optional] User ID of the super user or authorized Viewpoint user. The user ID of the super user is `dmc1_admin` and cannot be changed.

Required if security management is enabled on the Data Mover daemon.

Usage Notes

The following scenarios result in an error:

- A job is running when you execute the `restore_daemon` command
- You do not specify a directory where the backup files are stored
- You specify an absolute path, but `dm_user` does not have write privilege for the path
- You specify a relative or absolute path that does not exist
- The Data Mover repository being restored does not have the same hash algorithm as the Data Mover repository that was backed up by the `backup_daemon` command

XML File Example

To use an XML file such as `parameters.xml` with the `restore_daemon` command to restore previously backed up repository and configuration files of the Data Mover daemon, type `datamove restore_daemon -f parameters.xml`.

In the following example, `parameters.xml` specifies `repos_bu_001` as the directory under `daemon_backup` relative to the install location for the Data Mover daemon on your system where the previous backup was written.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<dmRestoreDaemon xmlns="http://schemas.teradata.com/dataMover/v2009//
  http://schemas.teradata.com/unity/datamover.xsd">
  <backup_target_dir>repos_bu_001</backup_target_dir>
</dmRestoreDaemon>
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="
```

save_configuration

Purpose

The `save_configuration` command saves information in the configuration file (that was created by the `list_configuration` command) to the Data Mover daemon using the `-f` parameter.

Syntax

[Data Mover XML Schemas](#)

Parameters

See [Parameter Order](#).

dm.rest.endpoint

[Optional] Enter a Data Mover REST server URL to overwrite the default value specified in the `commandline.properties` file in order to connect to a different REST server (and therefore a different daemon) at runtime.

`https://dm-server1:1443/datamover`

filename.xml

Name of the configuration XML file.

Example:

`configuration.xml`

The name of the configuration file by default is `configuration.xml`.

security_password

[Optional] Password for the super user or authorized Viewpoint user.

Example:

53cUr17y

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password_encrypted** is also specified.

security_password_encrypted

[Optional] Encrypted password for the super user.

Example:

052c7aabd14c7770141ac3c0137ab98ae0d3f0f7cddf588981206b010c0c1b2f

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password** is also specified.

security_username

[Optional] User ID of the super user or authorized Viewpoint user. The user ID of the super user is `dmc1_admin` and cannot be changed.

Required if security management is enabled on the Data Mover daemon.

XML File Example

For the `save_configuration` command, type `datamove save_configuration -f configuration.xml`.

The following is a sample `configuration.xml` file for use with the `save_configuration` command. When updating the configuration settings in the `configuration.xml` file, the number or text enclosed in the `<value>` element is the only area eligible for editing. The text in the `<description>` element provides additional informational only. Do not change the text enclosed in the key element.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<dmSaveConfiguration xmlns="http://schemas.teradata.com/dataMover/v2009"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://
schemas.teradata.com/unity/DataMover.xsd">
  <property>
    <key>agentCollector.agentHeartbeatWaitMillis</key>
    <value>600000</value>
    <description>Purpose: To set the amount of time to wait for an Agent
heartbeat before assuming it has gone out of service in milliseconds. Default:
600000.</description>
  </property>
  <property>
```



```

    <key>databaseQueryService.useBaseViewsOnly</key>
    <value>true</value>
    <description>Purpose: Set all data dictionary queries on Teradata source
and target systems to use the base views instead of X or VX views. Default: true.</
description>
  </property>
  <property>
    <key>different.session.charsets.enabled</key>
    <value>false</value>
    <description>Purpose: Whether or not the different session charsets feature
is supported. Default value false means not allowed.</description>
  </property>
  <property>
    <key>event.table.default</key>
    <value>NULL</value>
    <description>Purpose: Messages will be saved to this event table, unless
the messages come from a different event table or the Job Definition explicitly
overrides this parameter. Default: NULL.</description>
  </property>
  <property>
    <key>job.default.queryband</key>
    <value>ApplicationName=DM;Version=15.10;</value>
    <description>Purpose: Provide a set of name/value pairs for the default
query_band feature.</description>
  </property>
  <property>
    <key>job.default.queryband.enabled</key>
    <value>false</value>
    <description>Purpose: Enable/Disable the default queryband feature. Default
value: false</description>
  </property>
    <property>
      <key>job.databaseClientEncryption</key>
      <value>false</value>
      <description>Purpose: When set to true, utilities such as JDBC and TPTAPI
will initiate encrypted sessions to both the source and target database systems.
Default: false. Note: There is a performance hit trade-off for the gain of
encryption.</description>
    </property>
      <property>
        <key>job.force.direction</key>
        <value>false</value>
        <description>Purpose: To force direction of data movement from source to
target system.</description>

```

```

</property>
<property>
  <key>job.never.target.system</key>
  <value>>false</value>
  <description>Purpose: Prevent certain database systems from ever being a
target system in a Data Mover job. Default: false.</description>
</property>
<property>
  <key>job.noFallback</key>
  <value>>false</value>
  <description>Configures whether the fallback settings on tables created as
part of movement jobs are disabled. While this is enabled, fallback settings on
target tables are scrubbed and disabled. When this is disabled, fallback settings
will be set based on the source table. Default is false.</description>
</property>
<property>
  <key>job.onlineArchive</key>
  <value>>false</value>
  <description>Purpose: When set to true, online archiving is utilized for
objects that merit the use of DSA. Default: false. Note: There a is a performance
hit trade-off for the gain of object availability.</description>
</property>
<property>
  <key>job.overwriteExistingObjects</key>
  <value>>false</value>
  <description>Purpose: When set to true, objects that already exist on the
target database system are overwritten. Default: false.</description>
</property>
<property>
  <key>job.useSecurityMgmt</key>
  <value>>false</value>
  <description>Purpose: When set to true, some Data Mover commands will
require the admin username and password to be specified when executing the
command. Refer to the User Guide for a complete list of commands affected by this
parameter. Default: false</description>
</property>
<property>
  <key>job.useSyncService</key>
  <value>>false</value>
  <description>Purpose: To record any changes to the Data Mover repository
tables (inserts/updates/deletes) in an audit log table. The value must be set to
true in order to use the Sync service. Default: false.</description>
</property>
</property>

```

```

    <key>job.useUserIdPool</key>
    <value>>false</value>
    <description>Purpose: Use a target user from the pool of users. This enables
running multiple DSA tasks at the same time.</description>
  </property>
  <property>
    <key>repository.purge.definition.enabled</key>
    <value>>false</value>
    <description>Purpose: Enable/Disable the repository purge job definition
feature. Default value: false.</description>
  </property>
  <property>
    <key>repository.purge.enabled</key>
    <value>>true</value>
    <description>Purpose: Enable/Disable the repository purge feature.</
description>
  </property>
  <property>
    <key>repository.purge.history.unit</key>
    <value>days</value>
    <description>Purpose: The unit for job history data to kept in the
repository before purging should occur. The current supported values are days,
weeks, months, and years. Default value: days.</description>
  </property>
  <property>
    <key>repository.purge.history.unitcount</key>
    <value>60</value>
    <description>Purpose: The number of units for job history data to kept in
the repository before purging should occur. This value is combined with the value
for repository.purge.history.unit to determine the amount of time before purging
should occur for old jobs (for example, 60 days, 3 years, or 10 months). Default
value: 60. The value of -1 will disable the purging by time.</description>
  </property>
  <property>
    <key>repository.purge.hour</key>
    <value>1</value>
    <description>Purpose: The hour when the daily repository purging should
start. Default value 1 means 1am.</description>
  </property>
  <property>
    <key>repository.purge.minute</key>
    <value>0</value>
    <description>Purpose: The minute when the daily repository purging should
start. Default value 0.</description>

```

```

</property>
<property>
  <key>repository.purge.percent</key>
  <value>50</value>
  <description>Purpose: The percentage of repository permSPACE that needs to
be available to determine when purging should occur. Default value 50 means the
repository should be purged when more than 50% of the available permSPACE is in
use. The value of -1 will disable the purging by percentage.</description>
</property>
<property>
  <key>skip.FLML.tables</key>
  <value>true</value>
  <description>If enabled (true) and the table is being Fast/Multi Loaded,
the daemon marks it as COMPLETED WITH WARNINGS. If disabled (false) and the table
is being Fast/Multi Loaded, the daemon marks the table movement as FAILED. Default
is false.</description>
</property>
<property>
  <key>system.default.database.enabled</key>
  <value>false</value>
  <description>Purpose: Enable/Disable the default target/staging databases
at the system level. Default value false means disabled.</description>
</property>
<property>
  <key>target.system.load.slots</key>
  <value>5</value>
  <description>Purpose: Controls maximum number of load slots that Data Mover
can use at one time on target Teradata systems. Default: 5.</description>
</property>
<property>
  <key>tmsm.frequency.bytes</key>
  <value>2147483647</value>
  <description>Purpose: Controls frequency of sending messages when using
byte-based utilities (for example, TPTAPI). Default: 2147483647 bytes.</
description>
</property>
<property>
  <key>tmsm.mode</key>
  <value>NONE</value>
  <description>Purpose: Controls how Data Mover directs messages. When set
to BOTH, messages will be written to the TDI event tables. Default: NONE.</
description>

```

```
</property>
</dmSaveConfiguration>
```

start

The start command starts a job that was created with the create command. If the daemon does not have sufficient resources to run the job immediately, the job is queued.

Available parameters for this command may vary depending on the involved systems, the direction of data movement, and whether you are using the CLI or an .xml file. Refer to the parameter descriptions for guidance on the context in which you can use the parameters outlined in a section.

start

Purpose

The start command starts a job that was created with the create command. You may specify different job variable values than were originally used by entering them in the command line at runtime. You may also modify job variable values as well as the list of objects to copy by supplying an updated parameters.xml file. If the daemon does not have sufficient resources to run the job immediately, the job is queued.

Syntax

[Data Mover XML Schemas](#)

Parameters

See [Parameter Order](#).

data_streams

[Optional] Number of data streams to use between the source and target databases. Applies to jobs that use Teradata DSA and TPT API (to and from Teradata). All other protocols use a single data stream.

Example:

4

The default value is dynamically calculated by Data Mover.

db_client_encryption

[Optional] Set to true if job needs to be encrypted during data transfer.

dm.rest.endpoint

[Optional] Enter a Data Mover REST server URL to overwrite the default value specified in the commandline.properties file in order to connect to a different REST server (and therefore a different daemon) at runtime.

`https://dm-server1:1443/datamover`

force_utility

[Optional] Forces the Data Mover daemon to use a specific utility for all copy operations.

Valid Values

- dsa
- jdbc
- tptapi
- tptapi_load
- tptapi_stream
- tptapi_update
- T2T

If this value is not specified, the Data Mover daemon determines which Teradata utility is the best to use for the job.

Note:

Copying data to an older version of the Teradata Database using Teradata DSA is not valid. You cannot use Teradata DSA if the source and target TDPIIDs are the same.

Example:

dsa

job_name

Name of the job to be started.

Example:

12315DFHJKS

job_priority

[Optional] Specifies the execution priority for the job. Supported values are: HIGH, MEDIUM, and LOW, and UNSPECIFIED. If no value is specified, the default of MEDIUM is used at runtime.

Example:

MEDIUM

log_level

[Optional] Log level for log file output.

Valid Values

- 0

- 1
- 2
- 99

Example:

2

The default value is 0.

max_agents_per_task

[Optional] Maximum number of Data Mover agents to use in parallel when moving tables or databases.

Example:

4

The default value is dynamically calculated by Data Mover.

netrace

[Optional] CLI netrace parameter. Any value greater than or equal to 0 generates a CLI trace log. Valid CLI value must be provided.

netrace_buf_len

[Optional] CLI netrace_buf_len parameter. Any value greater than or equal to 0 generates a CLI trace log. Valid CLI value must be provided.

online_archive

[Optional] Allows read and write access to the source table(s) while the tables are being copied with Teradata DSA. Updates occur to the source table during the copy, but are not transferred to the target table. After a successful copy, the data contained in the target table matches the data that was in the source table at the beginning of the copy.

Valid Values

Value	Description
True	Enables online archive
False	Disables online archive
Unspecified	Default – the value is set to the value in the Data Mover daemon configuration file

Example:

true

overwrite_existing_objects

[Optional] Job overwrites objects that already exist on the target.

Valid Values

Value	Description
True	Enables overwriting
False	Disables overwriting
Unspecified	Default – the value is set to the value in the Data Mover daemon configuration file

If the parameter is not specified, the value is set to the `overwrite_existing_objects` parameter value in the Data Mover daemon configuration file. If the parameter is specified as true or false, that value takes precedence over the parameter value in the Data Mover daemon configuration file.

Example:

```
true
```

query_band

[Optional] A semicolon-separated set of name-value pairs that uniquely identifies Teradata sessions or transactions for the source and target. To use a query band to identify the job payroll, the user ID aa100000, and job session number 1122, define the query band as follows:

Example:

```
Job=payroll;Userid=aa100000;Jobsession=1122;
```

This parameter must be added as the last parameter in the XML job definition. See [About Query Band](#) for detailed syntax rules when defining a query band.

response_timeout

[Optional] Amount of time, in seconds, to wait for response from the Data Mover daemon.

Example:

```
60
```

save_changes

[Optional] Saves the changed job variable values and uses them to replace the values originally defined when the job was created.

security_password

[Optional] Password for the super user or authorized Viewpoint user.

Example:

53cUr17y

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password_encrypted** is also specified.

security_password_encrypted

[Optional] Encrypted password for the super user.

Example:

052c7aabd14c7770141ac3c0137ab98ae0d3f0f7cddf588981206b010c0c1b2f

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password** is also specified.

security_username

[Optional] User ID of the super user or authorized Viewpoint user. The user ID of the super user is `dmc1_admin` and cannot be changed.

Required if security management is enabled on the Data Mover daemon.

source_account_id

[Optional] Logon account ID for source database.

Note:

Spaces in the account name for the source or target account id causes the job to fail.

source_logon_mechanism

[Optional] Logon mechanism for source system. To log on to a source Teradata Database system, the user must provide at least one of the following:

- `source_user` *and* `source_password`
- `source_logon_mechanism`

Logon mechanisms are not supported for Teradata DSA jobs. Use logon mechanisms only for Teradata PT API and Teradata JDBC jobs. If **-source_logon_mechanism** is specified and **-force_utility** is not used, Teradata PT API is used by default. Specifying **-source_logon_mechanism** with Teradata DSA specified for **-force_utility** results in an error.

Example:

KRB5

source_logon_mechanism_data

[Optional] Additional parameters needed for the logon mechanism of the source system.

Example:

joe@domain1@@mypassword

source_password

[Optional] Source Teradata logon password.

Example:

123456789

Not a valid parameter if **-source_password_encrypted** is also specified. If you do not specify a password for this parameter, the command prompts you to enter it interactively. Input is masked with a set number of asterisks, regardless of the length of the password.

source_sessions

[Optional] Number of sessions per data stream on the source database.

Example:

4

The default value is dynamically calculated by Data Mover.

source_tdpid

[Optional] Source Teradata Database.

Example:

Checks

source_user

[Optional] Source Teradata logon id.

Example:

TD_API_user

If you do not specify a logon id for this parameter, the command prompts you to enter it interactively.

Note:

Spaces in the user name for the source or target id causes the job to fail.

sync

[Optional] Waits for the job to complete, then returns an exit code that indicates if the job completed successfully. An exit code of 0 indicates successful job completion. An exit code other than 0 indicates an error with the job or with the command.

target_account_id

[Optional] Logon account ID for target database.

Note:

Spaces in the account name for the source or target account id causes the job to fail.

target_logon_mechanism

[Optional] Logon mechanism for target system. To log on to a target Teradata Database system, the user must provide at least one of the following:

- `target_user` *and* `target_password`
- `target_logon_mechanism`

Teradata DSA does not support logon mechanisms. Use logon mechanisms only with Teradata PT API and Teradata JDBC jobs. If **-target_logon_mechanism** is specified and **-force_utility** is not used, Teradata PT API is used by default. Specifying **-target_logon_mechanism** with Teradata DSA specified for **-force_utility** results in an error.

Example:

KRB5

target_password

[Optional] Target Teradata logon password.

Example:

212133344

Not a valid parameter if **-target_password_encrypted** is also specified. If you do not specify a password for this parameter, the command prompts you to enter it interactively. Input is masked with a set number of asterisks, regardless of the length of the password.

target_sessions

[Optional] Number of sessions per data stream on the target database.

Example:

4

The default value is dynamically calculated by Data Mover.

target_tdpid

[Optional] Target Teradata Database.

Example:

Leo

target_user

[Optional] Target Teradata logon id.

Example:

TD_tar_User

If you do not specify a logon id for this parameter, the command prompts you to enter it interactively.

Note:

Spaces in the user name for the source or target id causes the job to fail.

tpt_debug

[Optional] TPT API trace debug log parameter. Any value greater than or equal to 0 generates a TPT API trace log. Valid TPT API value must be provided.

uowid

[Optional] Alternate ID or name for the batch of work associated with the job. If you provide a value for this parameter, Data Mover reports this value as the unit of work ID when sending events to Teradata Ecosystem Manager or to its internal TMSMEVENT table. If you do not specify this parameter, Data Mover uses a default value as the unit of work ID when sending events to Teradata Ecosystem Manager or to its internal TMSMEVENT table. The default value for the unit of work ID is composed of the job execution name and the current timestamp. For example, if you want to define the origins of a query source the job execution name is `sales_table`, the default value of the unit of work ID is `sales_table-20211110122330`

Example:

`sales_tables_start`

Usage Notes

Each time a job starts, a new job instance is created with the name *job name-date time year*. This name appears in the status command or the logs to identify information for this specific execution of the job. Use the status command to monitor the job status after the job has been started.

If an instance of a job has not completed all specified steps, a new instance of the job cannot be started. If the daemon does not have adequate resources to run the job immediately, the job is placed in a queue. Copying the same objects to the same target as another job that is running or queued is not allowed.

You can supply new values for job variables at runtime using either of the following two methods:

- Specify the value for the job variable directly in the command line, just as you can using the create command. For example to set the logging level for job1 to a value of 99, type: `datamove start -job_name job1 -log_level 99`.
- Specify the new value for the job variable by modifying the XML file and submitting it to the start command, just as you can using the create command: `datamove start -job_name job1 -f job1.xml`

If you want to modify the list of objects to copy, you must do so by modifying and submitting the XML file. Be sure that the XML file contains all of the objects you want to copy, not only those with name changes. If an object that was originally specified to be copied is no longer included in the list, it is not included in the job execution.

By default, any job variables supplied at runtime are not saved to the job definition. Therefore, if you run the job again, the job reverts to the original settings. You can save the new job variable values for future use by using the save changes parameter in either the command line or file. For example:

- At the command line prompt, type: `datamove start -job_name job1 -log_level 99 -save_changes`
- In the XML file, specify: `<saveChanges>true</saveChanges>`

Note:

The save changes parameter syntax is different when used at the command line versus in an XML file, as shown in the example.

If save changes is enabled, the modified job definition is saved as the new base job definition. The modification of job variable values triggers a rebuild of the job plan, overriding the freeze job steps option, if enabled. The new job plan is then used for the current run. If save changes is *not* enabled, the base job definition is not changed. The job plan is rebuilt and the modified variables are used for the current run, but the base job definition remains unchanged and all subsequent executions use the base job definition.

Security settings impact certain start command options in several ways:

- If security is enabled, a user must have write permission to be able to set the save changes parameter to true.
- A user who is not the job owner must supply source and target Teradata system user names and passwords when specifying the object list in the XML and submitting it using the start command.
- If security is enabled and **job_security** is specified in the modified XML to change job permission, the user must be `dcml_admin` or the job owner, and the user must provide all the permissions, not just the modified permissions. If **job_owner** is specified in **job_security** and the user wants to change the job owner, the user must be `dcml_admin`.

XML File Example

For the start command, type `datamove start -f parameters.xml`.

In the following example, the XML file shown defines the job `testStart` which copies tables `fmt_inf` and `test 1` to the target using DSA.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<dmCreate xmlns="http://schemas.teradata.com/dataMover/v2009"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schemas.teradata.com/unity/datamover.xsd">
  <job_name>testStart</job_name>
  <source_tdpid>dmdev</source_tdpid>
  <source_user>dmuser</source_user>
  <source_password>dbc</source_password>
  <target_tdpid>dm-daemon2</target_tdpid>
  <target_user>dmuser</target_user>
  <target_password>dbc</target_password>
  <data_streams>1</data_streams>
  <source_sessions>1</source_sessions>
  <target_sessions>1</target_sessions>
  <overwrite_existing_objects>TRUE</overwrite_existing_objects>
  <freeze_job_steps>true</freeze_job_steps>
  <force_utility>DSA</force_utility>
  <log_level>1</log_level>
  <online_archive>false</online_archive>
  <database selection="unselected">
    <name>testdb</name>
    <table selection="included">
      <name>fmt_inf</name>
      <validate_row_count>ALL</validate_row_count>
    </table>
    <table selection="included">
      <name>test1</name>
      <compare_ddl>true</compare_ddl>
    </table>
  </database>
</dmCreate>
```

In the next example, the XML file has been modified and is called `changedParameters.xml`. You could run `start -job_name testStart -f changedParameters.xml - force_utility tptapi -sync` to run a job that copies only table `test3` and records where `i = 2` from table `fmt_inf` to the target using TPTAPI.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<dmEdit xmlns="http://schemas.teradata.com/dataMover/v2009"
```

```

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schemas.teradata.com/unity/DataMover.xsd">
  <job_name>testStart</job_name>
  <source_user>dmuser</source_user>
  <source_password>dbc</source_password>
  <target_user>dmuser</target_user>
  <target_password>dbc</target_password>
  <database selection="unselected">
    <name>testdb</name>
    <table selection="included">
      <name>fmt_inf</name>
      <validate_row_count>ALL</validate_row_count>
      <compare_ddl>true</compare_ddl>
      <sql_where_clause>
        <![CDATA[ where i = 2]]>
      </sql_where_clause>
      <key_columns>
        <key_column>i</key_column>
      </key_columns>
    </table>
    <table selection="included">
      <name>test3</name>
      <validate_row_count>ALL</validate_row_count>
      <compare_ddl>true</compare_ddl>
    </table>
  </database>
</dmEdit>

```

status

Purpose

The status command displays the status of a specified job. To see the status of multiple jobs, issue a command for each job.

The frequency of status updates can also be specified. If the frequency parameter is not specified, the status is displayed only once and is not updated.

Syntax

[Data Mover XML Schemas](#)

Parameters

See [Parameter Order](#).

dm.rest.endpoint

[Optional] Enter a Data Mover REST server URL to overwrite the default value specified in the `commandline.properties` file in order to connect to a different REST server (and therefore a different daemon) at runtime.

`https://dm-server1:1443/datamover`

frequency

[Optional] Frequency of status update in seconds.

Example:

5

job_name

Name of the job to be monitored.

Example:

12315DFHJKS

output_level

[Optional] Output level of the job status. Specify one of the output levels in the following table:

Output Level	Description
1	Displays the overall job status. This is the default.
2	Displays the status of each job step.
3	Displays the status of each job task.
4	Displays detailed log information.

Example:

2

security_password

[Optional] Password for the super user or authorized Viewpoint user.

Example:

53cUr17y

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password_encrypted** is also specified.

security_password_encrypted

[Optional] Encrypted password for the super user.

Example:

052c7aabd14c7770141ac3c0137ab98ae0d3f0f7cddf588981206b010c0c1b2f

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password** is also specified.

security_username

[Optional] User ID of the super user or authorized Viewpoint user. The user ID of the super user is dmc1_admin and cannot be changed.

Required if security management is enabled on the Data Mover daemon.

Usage Notes

To stop the status updates, press the system INTR character (UNIX) or Ctrl+C (Windows).

The following example shows output for the status command.

```
Job Name: floyd_to_checks-Sun_May_24_13:42:13_PDT_2009
Job Execution Name:
floyd_to_checks-
Sun_May_24_13:42:13_PDT_2009_Execution_Sun_May_24_13:48:40_PDT_2009_1

TYPE ID STATUS CURRENT STEP START TIME DURATION TIME
-----
Job: 3 COMPLETED_SUCCESSFULLY - 5/24/09 1:49 PM 0:0:55

TYPE ID STATUS STEP TYPE START TIME DURATION TIME
-----
Step: 1 COMPLETED_SUCCESSFULLY MOVE_DEFINITION_BEFORE_LOAD 5/24/09 1:49 PM 0:0:7
Step: 2 COMPLETED_SUCCESSFULLY MOVE_TABLE_DATA 5/24/09 1:49 PM 0:0:28
Step: 3 COMPLETED_SUCCESSFULLY RESOLVE_TABLE_AFTER_LOAD 5/24/09 1:49 PM 0:0:6
```

XML File Example

For the status command, type `datamove status -f parameters.xml`.

In the following example, the parameters file returns the status of each job step for job 12315DFHJKS, and updates the status every five seconds.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<dmStatus xmlns="http://schemas.teradata.com/dataMover/v2009"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://schemas.teradata.com/unity/DataMover.xsd">
  <job_name>12315DFHJKS</job_name>
  <frequency>5</frequency>
```

```
<output_level>2</output_level>
</dmStatus>
```

stop

Purpose

The stop command stops a running job.

If the job is running, the daemon finishes the current unit of work, then stops at the earliest convenient time.

If the job is waiting in the job queue, the job is dropped from the queue.

Syntax

[Data Mover XML Schemas](#)

Parameters

See [Parameter Order](#).

dm.rest.endpoint

[Optional] Enter a Data Mover REST server URL to overwrite the default value specified in the `commandline.properties` file in order to connect to a different REST server (and therefore a different daemon) at runtime.

`https://dm-server1:1443/datamover`

job_name

Name of the job to be stopped.

Example:

`12315DFHJKS`

security_password

[Optional] Password for the super user or authorized Viewpoint user.

Example:

`53cUr17y`

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password_encrypted** is also specified.

security_password_encrypted

[Optional] Encrypted password for the super user.

Example:

`052c7aabd14c7770141ac3c0137ab98ae0d3f0f7cddf588981206b010c0c1b2f`

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password** is also specified.

security_username

[Optional] User ID of the super user or authorized Viewpoint user. The user ID of the super user is `dmcl_admin` and cannot be changed.

Required if security management is enabled on the Data Mover daemon.

XML File Example

For the stop command, type `datamove stop -f parameters.xml`.

In the following example, the parameters file stops job 12315DFHJKS.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<dmStop
  xmlns="http://schemas.teradata.com/dataMover/v2009"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://schemas.teradata.com/unity/DataMover.xsd">
  <job_name>12315DFHJKS</job_name>
</dmStop>
```

update_job_priorities

Purpose

The command updates the priority of one or more jobs in queue using only the `parameters.xml` file, not directly in the command line.

Syntax

[Data Mover XML Schemas](#)

Parameters

See [Parameter Order](#).

job_name

Name of the job for which you want to update the priority.

Example:

test1

job_priority

New priority for the job.

Example:

HIGH

security_password

[Optional] Password for the super user or authorized Viewpoint user.

Example:

53cUr17y

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password_encrypted** is also specified.

security_password_encrypted

[Optional] Encrypted password for the super user.

Example:

052c7aabd14c7770141ac3c0137ab98ae0d3f0f7cddf588981206b010c0c1b2f

Required if security management is enabled on the Data Mover daemon. Not a valid parameter if **-security_password** is also specified.

security_username

[Optional] User ID of the super user or authorized Viewpoint user. The user ID of the super user is `dmc1_admin` and cannot be changed.

Required if security management is enabled on the Data Mover daemon.

XML File Example

For the `update_job_priorities` command, type `datamove update_job_priorities -f parameters.xml`.

In the following example, the parameters file changes the priority for jobs `test1` and `test2`.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<dmUpdateJobPriorities xmlns="http://schemas.teradata.com/dataMover/v2009"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schemas.teradata.com/unity/DataMover.xsd">
  <job_priority_set>
    <job_name>test1</job_name>
    <job_priority>LOW</job_priority>
  </job_priority_set>
  <job_priority_set>
    <job_name>test2</job_name>
    <job_priority>HIGH</job_priority>
  </job_priority_set>
</dmUpdateJobPriorities>
```

```
</job_priority_set>
</dmUpdateJobPriorities>
```

update_job_steps

Purpose

The `update_job_steps` command updates the steps for the job you specify without starting or running the job. This command is very helpful if you specified the `freeze_job_steps` parameter as `true` when you created the job, and subsequently there were some changes in the source or target environments.

Syntax

[Data Mover XML Schemas](#)

Parameters

See [Parameter Order](#).

dm.rest.endpoint

[Optional] Enter a Data Mover REST server URL to overwrite the default value specified in the `commandline.properties` file in order to connect to a different REST server (and therefore a different daemon) at runtime.

`https://dm-server1:1443/datamover`

job_name

Name of the job.

Example:

`12333KZIHRT`

RESTful API

The Data Mover RESTful API

You can use the Data Mover RESTful API to create, run, and monitor jobs using any standard REST client.

For information on Data Mover RESTful API, see [Data Mover RESTful API](#).

RESTful API Status Codes

One of the following HTTP status codes is returned in response to a RESTful API call:

Code	Definition	Description
200	OK	Request has succeeded.
201	Created	Request has been fulfilled and a new resource is being created.
301	Moved Permanently	Requested URL has been moved permanently and should be redirected. User the 'location' header to redirect.
400	Bad Request	Request could not be understood by the server due to malformed syntax.
401	Unauthorized	Request requires user authentication.
404	Not Found	Resource referenced by the specified URL was not found.
409	Conflict	Request could not be completed due to a conflict with the current state of the resource.
412	Precondition Failed	Request could not be completed due to invalid input parameter.
500	Internal Server Error	Data Mover service encountered an unexpected condition that prevented it from fulfilling the request.

Note:

If the API returns an error code (codes 400–500), the response body will contain a response parameter message with an explanation of the error.

RESTful API Security

The authentication and security of the REST container depends on security management of the daemon.

Status	Description
Disabled on Data Mover Daemon	The REST client (user) does not have to provide a username and password if Data Mover daemon security is disabled.
Enabled on Data Mover Daemon	<p>The REST client (user) must provide the Viewpoint username and password if Data Mover daemon security is enabled. The Data Mover Daemon leverages Viewpoint for authentication and authorization.</p> <p>It is recommended that the HTTPS protocol is used for security purposes if Viewpoint credentials are provided to Data Mover. To enable HTTPS in the REST container, see the <i>Teradata® Query Service Installation, Configuration, and Upgrade Guide for Customers</i>.</p> <p>The Viewpoint credentials for Data Mover requests must pass as basic access authentication by the client.</p> <p>To use a Data Mover encrypted password when using REST, set Password-Encrypted to true in the header. If value is false or not included in the header, the authorization is treated as a plain text password.</p>

RESTful APIs

Backup Daemon (RESTful API)

Purpose

Back up the Data Mover daemon repository and configuration files to a directory using the following URL and method:

Item	Description
URL	/datamover/backups
Method	POST

Request Parameters

directory

Description: Directory name to store the backup

JSON Data Type: String

Required: No

Note:

Default directory /var/opt/teradata/datamover/daemon_backup. If specifying a directory, the path must be relative to the default path or an absolute path pointing to a directory that dmuser has access.

Response Parameters

message

Description: States if the backup was successful or failed

JSON Data Type: String

Response Example

```
{
  "message": "Daemon backup job has finished, backup files is saved at
location : /var/opt/teradata/datamover/daemon_backup/test1/."
}
```

Check Permissions (RESTful API)

Overview

Check if user and roles specified in the header have the necessary permissions for jobs, daemon advanced settings, and daemon access using the following URL and method:

Item	Description
URL	/datamover/permissions
Method	POST

Note:

When security is enabled, only a Viewpoint or command line admin can check daemon access and daemon advanced permissions. Non-admin users can only check job permissions.

Request Header

Authorization

Description: Basic header authentication

Note:

User must be **dmcl_admin** if call is from Viewpoint.

JSON Data Type: String

Required: Yes

Portlet-User

Description: Viewpoint user login

JSON Data Type: String

Required: No, unless call is made from Viewpoint

Portlet-Roles

Description: Roles associated with Viewpoint user login

JSON Data Type: String, separated by commas when more than one role exists

Required: No, unless call is made from Viewpoint

Request Parameters**logic (not currently available)**

Description: What logic to use to combine permissions when multiple permissions exist

JSON Data Type: String, valid values are and or or

Required: No

permissions

Description: A list of permissions or actions the user or roles currently has

Note:

Currently only one permission is allowed in the list.

JSON Data Type: String

Required: Yes

Response Parameters

No response parameters required.

Examples

The following example checks if a user or role specified in the header has `daemon_access` permissions:

```
{
  "enumerate": true,
  "permissions": ["datamover:daemon_access"]
}
```

The following example lists a true response for a user or role having `daemon_access` permissions:

```
{ "result" : true,
  "permissions" : [ {
                        "resource" : "tdrn:datamover:daemon_access:*",
                        "actions" : [ "read", "write", "execute" ]
                      } ]
}
```

Clean Up Job (RESTful API)

Purpose

Clean up or delete an executed job using the following URL and method:

Item	Description
URL	/datamover/executedJobs/ <i>job-executed-name</i> ?cleanupOnly=true
Method	DELETE

Request Parameters

cleanupOnly

Description: Flag to run cleanup

Required: Yes

Response Parameters

No response parameters required.

Response Examples

```
{
  "error": "invalid credential/not authorized to perform the delete action"
}
```

```
{
  "error": "Job [job-executed-name] was not deleted because it is
```

```
still running"
}
```

Related Information:

[Viewing Data Mover REST Samples](#)
[RESTful API Status Codes](#)
[RESTful API Security](#)
[RESTful API Object Types](#)

Create a Cloud Staging Copy Job (RESTful API)

Overview

You need a valid cloud staging area for creating a cloud staging copy Job. To create a cloud staging copy Job add a **cloudStagingArea** object inside the settings object of the request JSON. Make sure that the object specifies a valid cloud staging area for the job.

Create a cloud staging copy job using the following URL and method:

Item	Description
URL	/datamover/jobs
Method	POST

Request Parameters

jobName

Description: Name of the job
 JSON Data Type: String
 Required: No

uowind

Description: Unit of work id
 JSON Data Type: String
 Required: No

sourceLogin

Description: Properties of the source system
 JSON Data Type: Object ([LoginType](#))
 Required: Yes

targetLogin

Description: Properties of the target system

JSON Data Type: Object ([LoginType](#))

Required: Yes

settings

Description: Properties of the job configuration

JSON Data Type: Object ([SettingsType](#))

Required: No

jobSecurity

Description: Job security details

JSON Data Type: Object ([JobSecurityType](#))

Required: No

objects

Description: Objects to be copied

JSON Data Type: Object ([DBObjectType](#))

Required: Yes

Response Parameters

No response parameters required.

Response Example

```
{
  "error": "job creation failed, table does not exist in source database"
}
```

Request Example

```
{
  "jobName": "DMCSJob",
  "sourceLogin": {
    "teradata": {
      "username": "user",
      "password": "password",
      "tdpid": "sourceSystem",

```

```

        "sessionCharset": "UTF8",
        "passwordEncrypted": "false"
    },
    },
    "targetLogin": {
        "teradata": {
            "username": "user",
            "password": "password",
            "tdpid": "targetSystem"
        },
        "useTargetUserIdPool": false
    },
    "settings": {
        "priority": "MEDIUM",
        "overwriteExistingObjects": "true",
        "freezeJobSteps": "false",
        "targetDatabase": "targetDatabaseJobLevel",
        "compareDDL": "true",
        "logLevel": "99",
        "tdTdSettings":
        {
            "forceUtility": "DSA",
            "dataStreams": 5,
            "sourceSessions": 10,
            "targetSessions": 10,
            "onlineArchive": "false"
        },
        "enableTraceLog": {
            "cliTraceLog": {
                "netrace": -1,
                "netraceBufLen": -1
            },
            "tptTraceLog": {
                "tptapiDebug": -1
            }
        },
        "cloudStagingArea": {
            "name": "CSArea"
        },
        "enableIncrementalRestore": "FALSE"
    },
    "objects": {
        "database": [{
            "name": "dm19948",

```

```

    "selection": "UNSELECTED",
    "table": [{
      "name": "testtable",
      "ownerName": "devuser",
      "useSourceStagingTable": false,
      "forceTargetStagingTable": false,
      "targetDatabase": "test",
      "targetName": "test",
      "validateRowCount": "NONE",
      "selection": "INCLUDED",
      "teradataParameters": {
        "allowTPTLoadMultiset": false,
        "overrideLockAccess": false
      }
    }]
  }
}

```

Create Cloud Staging Area (RESTful API)

Overview

Following are the two options for creating a Cloud Staging Area:

- Provide Source and Target system Target Groups that have already been defined in DSC and paired in a Target Group Map.
- Provide AWS S3 information and allow Data Mover to define the Target Groups and Map.

Create a Cloud Staging Area using the following URL and method:

Item	Description
URL	/datamover/cloud-staging-areas
Method	POST

Request Parameters

name

Description: Name of the cloud staging area

JSON Data Type: String

Required: Yes

storage_type

Description: Specify cloud storage type. Currently only S3 is supported

JSON Data Type: String

Required: Yes

s3_properties

Description: AWS S3 information used to create the cloud staging area

JSON Data Type: Object ([S3Properties](#))

Required: No

source_target_pairs

Description: Source and target system information

JSON Data Type: JSON Array ([SourceTargetPair](#))

Required: Yes

Response Parameters**account_name**

Description: Name of the created cloud staging area

JSON Data Type: String

messages

Description: Contains error and warning messages

JSON Data Type: Object ([Messages](#))

Response Example

```
{
  "messages": {
    "errors": [],
    "warnings": []
  },
  "account_name": "my_staging_area"
}
```

The following is a request example to create cloud staging area by providing predefined Target Groups:

```
{
  "name": "cs2-cloudstagingarea",
  "storage_type": "S3",
  "source_target_pairs": [
    {
      "source_system": "sourceSystem",
      "source_system_target_group": "sourceTGroup",
      "target_system": "targetSystem",
      "target_system_target_group": "targetTGroup"
    },
    {
      "source_system": "targetSystem2",
      "source_system_target_group": "targetTGroup2",
      "target_system": "sourceSystem2",
      "target_system_target_group": "sourceTGroup2"
    }
  ]
}
```

The following is a request example to create cloud staging area by providing AWS information where Data Mover creates Target Groups:

```
{
  "name": "cs2-cloudstagingarea",
  "storage_type": "S3",
  "s3_properties": {
    "access_key_id": "ABCDEFGH",
    "secret_access_key": "AbcDEfgHIjklmNop123/456qRstUVwXyZ",
    "buckets_by_region": [
      {
        "buckets": [
          {
            "bucket_name": "example-bucket",
            "prefix_list": [
              {
                "prefix_name": "backup",
                "storage_devices": 100
              }
            ]
          }
        ],
        "region": "us-west-2"
      }
    ]
  }
}
```



```

    },
    "source_target_pairs": [
      {
        "source_system": "sourceSystem",
        "target_system": "targetSystem"
      },
      {
        "source_system": "sourceSystem2",
        "target_system": "targetSystem2"
      }
    ]
  }
}

```

Create Event Table (RESTful API)

Overview

Create a new event table using the following URL and method:

Item	Description
URL	/datamover/event-tables
Method	POST

Request Parameters

event_table_id

Description: Unique ID for the event table

JSON Data Type: String

Required: Yes

event_table_host

Description: Name of the external Teradata host to place the event table

JSON Data Type: String

Required: Yes

event_table_parent_database

Description: Name of the parent database where the TMSMEvent table is to be created

JSON Data Type: String

Required: Yes

host_user_name

Description: Teradata user name used to access **event_table_host** and create the TMSMEvent table. This user name is also used to access the event table when inserting events.

JSON Data Type: String

Required: Yes

host_user_password

Description: Password for **host_user_name**

JSON Data Type: String

Required: Yes

use_existing_event_table

Description: Name of the existing TMSMEvent on host in parent database to use to create a new event table.

JSON Data Type: String

Required: No

Response Parameters**event_table_id**

Description: The same event table ID provided in the request body

JSON Data Type: String

message

Description: Message explaining success or failure

JSON Data Type: String

Response Examples

```
{
  "event_table_id" : "test1",
  "message" : "Created Table st_stg_dm_user.TMSMEVENT on system td1 "
}
```

```
{
  "message" : "Must provide event_table_id in JSON Body Request."
}
```

```
{
  "event_table_id" : "event1",
  "message" : "Error: problem querying system td1 for existing tmsmEvent
in database DM1 Cause: Error creating data source for td1/event1.
Cause: Error while creating data source for td1/event1/Cannot create
PoolableConnectionFactory ([Teradata Database] [TeraJDBC 16.20.00.10] [Error
8017] [SQLState 28000] The UserId, Password or Account is invalid.) "
}
```

Create Job (RESTful API)

Purpose

Create a job using the following URL and method:

Item	Description
URL	/datamover/jobs
Method	POST

Request Parameters

jobName

Description: Name of the job

JSON Data Type: String

Required: No

uowid

Description: Unit of work id

JSON Data Type: String

Required: No

sourceLogin

Description: Properties of the source system

JSON Data Type: Object ([LoginType](#))

Required: Yes

targetLogin

Description: Properties of the target system

JSON Data Type: Object ([LoginType](#))

Required: Yes

settings

Description: Properties of the job configuration

JSON Data Type: Object ([SettingsType](#))

Required: No

jobSecurity

Description: Job security details

JSON Data Type: Object ([JobSecurityType](#))

Required: No

objects

Description: Objects to be copied

JSON Data Type: Object ([DbObjectType](#))

Required: Yes

Response Parameters

No response parameters required.

Response Example

```
{
  "error": "job creation failed, table does not exist in source database"
}
```

Related Information:

[Viewing Data Mover REST Samples](#)

[RESTful API Status Codes](#)

[RESTful API Security](#)

[RESTful API Object Types](#)

Create Policies (RESTful API)

Overview

Create permissions needed for jobs, daemon advanced settings, and daemon access using the following URL and method:

Item	Description
URL	/datamover/policies
Method	POST

Note:

When security is enabled, only a Viewpoint or command line super user can create policies.

Request Header

Authorization

Description: Basic header authentication

Note:

User must be **dmcl_admin** if call is from Viewpoint.

JSON Data Type: String

Required: No, unless **securityMgmt** is enabled.

Portlet-User

Description: Viewpoint user login

JSON Data Type: String

Required: No, unless call is made from Viewpoint

Portlet-Roles

Description: Roles associated with Viewpoint user login

JSON Data Type: String, separated by commas when more than one role exists

Required: No, unless call is made from Viewpoint

Request Parameters

policies

Description: A list of objects which associate a set of permissions to a set of users or roles

JSON Data Type: String

Required: Yes

Response Parameters

No response parameters required.

Examples

The following is a request example to create policies:

```
[ {
  "service" : "datamover",
  "type" : "user",
  "principals" : [ "admin", "user1"],
  "actions" : [ "read", "write", "execute" ],
  "resources" : [ "tdrn:datamover:job:test",
"tdrn:datamover:job:testCP_002", "tdrn:datamover:job:dm31111_001" ]
}, {
  "service" : "datamover",
  "type" : "user",
  "principals" : [ "user1" ],
  "actions" : [ "owner" ],
  "resources" : [ "tdrn:datamover:job:test", "tdrn:datamover:job:testCP_002" ]
},{
  "service" : "datamover",
  "type" : "user",
  "principals" : [ "tester_002"],
  "actions" : [ "owner" ],
  "resources" : [ "tdrn:datamover:job:dm31111_001" ]
}
```

```
]

```

The following failed response code example is when the user does not have permissions to create or replace policies:

```
{ "message" : "When security is on, only commandline super user or viewpoint
could create policies. The user does not have the permission to run
UPDATE_JOB_PERMISSIONS command" }
```

Delete Cloud Staging Area (RESTful API)

Overview

Delete a Cloud Staging Area using the following URL and method:

Item	Description
URL	/datamover/cloud-staging-areas/staging_area_name
Method	DELETE

Request Parameters

No request parameters required.

Response Parameters

messages

Description: Contains error and warning messages

JSON Data Type: Object ([Messages](#))

Response Example

```
{
  "messages": {
    "errors": [],
    "warnings": []
  }
}
```

Delete Event Table (RESTful API)

Overview

Delete an existing event table using the following URL and method:

Item	Description
URL	/datamover/event-tables/ <i>event-table-id</i>
Method	DELETE

Request Parameters

No specific request parameters required.

Response Parameters

message

Description: Message explaining success or failure

JSON Data Type: String

Response Examples

```
{
  "message" : "Dropped Event Table st_stg_dm_user.TMSMEVENT from system td1.
Deleted Stored Event Table Information "
}
```

```
{
  "message" : "Error: Event Table With event_table_name event1 Not Found "
}
```

Delete Job (RESTful API)

Purpose

Data Mover provides the following options when deleting jobs:

- Delete a job definition and all instances of the job
- Delete only the job instances without deleting the job definition

The following URL and methods are available for deleting jobs:

Item	Description
URL	<code>/datamover/jobs/job-name</code> <code>/datamover/jobs/job-name?definition=value</code>
Method	DELETE

Request Parameters

definition

Description: Flag to delete job instance only without job definition

Required: No

Accepted Values: True or False

Note:

Default value is true when no value is entered.

- True = delete job definition and job instance
- False = delete only job instance

Response Parameters

No response parameters required.

Response Examples

```
{
  "error": "job not found"
}
```

```
{
  "error": "Job Definition [job-name] was not deleted because an instance of
```

```
it is still running"
}
```

Related Information:

[Viewing Data Mover REST Samples](#)
[RESTful API Status Codes](#)
[RESTful API Security](#)
[RESTful API Object Types](#)

Edit Cloud Staging Area (RESTful API)

Overview

This API allows you to edit an existing cloud staging area. Limitations to what you can edit using the API in an existing cloud staging area are:

- Access Key ID (Only if initially provided when creating the cloud staging area.)
- Secret Access Key (Only if initially provided when creating the cloud staging area.)
- Source Target Pairs.

The properties of the cloud staging area you cannot edit are:

- Cloud Staging Area name
- Buckets by Region

To change the name of the cloud staging area or the Buckets by Region properties, you have to delete the cloud staging area and create again with the desired changes.

Edit the existing cloud staging area using the following URL and method:

Item	Description
URL	/datamover/cloud-staging-areas/staging_area_name
Method	PUT

Request Parameters

name

Description: Name of the cloud staging area

JSON Data Type: String

Required: Yes

storage_type

Description: Specify cloud storage type. Currently only S3 is supported

JSON Data Type: String

Required: Yes

s3_properties

Description: AWS S3 information used to create the cloud staging area

JSON Data Type: Object ([S3Properties](#))

Required: No

source_target_pairs

Description: Source and target system information

JSON Data Type: JSON Array ([SourceTargetPair](#))

Required: Yes

Response Parameters

account_name

Description: Name of the created cloud staging area

JSON Data Type: String

messages

Description: Contains error and warning messages

JSON Data Type: Object ([Messages](#))

Response Example

```
{
  "messages": {
    "errors": [],
    "warnings": []
  },
  "account_name": "my_staging_area"
}
```

The following is a request example to edit the cloud staging area that was initially created with predefined target groups and target group mappings. In this scenario you can modify only **source_target_pairs**.

```
{
  "name": "cs2-cloudstagingarea",
  "storage_type": "S3",
  "source_target_pairs": [
```

```

    {
      "source_system": "sourceSystem",
      "source_system_target_group": "sourceTGroup",
      "target_system": "targetSystem",
      "target_system_target_group": "targetTGroup"
    },
    {
      "source_system": "targetSystem2",
      "source_system_target_group": "targetTGroup2",
      "target_system": "sourceSystem2",
      "target_system_target_group": "sourceTGroup2"
    }
  ]
}

```

The following is a request example to edit a cloud staging area where Data Mover configured DSC for AWS S3, target groups, and target group mappings. In this scenario you can modify the **access_key_id**, **secret_access_key**, and **source_target_pairs**.

```

{
  "name": "cs2-cloudstagingarea",
  "storage_type": "S3",
  "s3_properties": {
    "access_key_id": "ABCDEFGH",
    "secret_access_key": "AbcDEfgHIjklmNop123/456qRstUVwXyZ",
    "buckets_by_region": [
      {
        "buckets": [
          {
            "bucket_name": "example-bucket",
            "prefix_list": [
              {
                "prefix_name": "backup",
                "storage_devices": 100
              }
            ]
          }
        ]
      },
      {
        "region": "us-west-2"
      }
    ]
  },
  "source_target_pairs": [
    {

```

```

        "source_system": "sourceSystem",
        "target_system": "targetSystem"
    },
    {
        "source_system": "sourceSystem2",
        "target_system": "targetSystem2"
    }
]
}

```

Edit Event Table (RESTful API)

Overview

Edit the username and password for an existing event table using the following URL and method:

Item	Description
URL	/datamover/event-tables/ <i>event-table-id</i>
Method	PUT

Request Parameters

host_user_name

Description: New Teradata username to use for accessing the event table

JSON Data Type: String

Required: No

host_user_password

Description: New password to use for accessing the event table

JSON Data Type: String

Required: No

Response Parameters

message

Description: Message explaining success or failure

JSON Data Type: String

Response Examples

```
{
  "message" : "Updated password for User st_stg_dm_user. Updated User Name to
st_stg_dm_user2 "
}
```

```
{
  "message" : "Must provide host_user_name and/or host_user_password."
}
```

```
{
  "message" : "Error: Event Table With event_table_name dummy Not Found "
}
```

Edit Job (RESTful API)

Purpose

Edit a job using the following URL and method:

Item	Description
URL	/datamover/jobs/job-name
Method	PUT

Request Parameters

uowid

Description: Unit of work ID

JSON Data Type: String

Required: No

sourceLogin

Description: Source system properties

JSON Data Type: Object ([LoginType](#))

Required: No

targetLogin

Description: Target system properties

JSON Data Type: Object ([LoginType](#))

Required: No

settings

Description: Job configuration properties

JSON Data Type: Object ([SettingsType](#))

Required: No

jobSecurity

Description: Job security details

JSON Data Type: Object ([JobSecurityType](#))

Required: No

objects

Description: Objects to be copied

JSON Data Type: Object ([DbObjectType](#))

Required: No

Response Parameters

No response parameters required.

Response Example

```
{
  "message" : "Job definition not found: failed-multi-table"
}
```

edit RESTful API Example (Teradata to Teradata)

```
{
  "sourceLogin":
  {
    "teradata":
    {
      "tdpid": "sourceSystem",
      "username": "user",
      "password": "password",
      "sessionCharset": "UTF8",
      "passwordEncrypted": "false"
    }
  }
}
```

```

},
"targetLogin":
{
  "userPool": "poolA"
},
"settings":
{
  "priority": "MEDIUM",
  "overwriteExistingObjects": "true",
  "freezeJobSteps": "false",
  "targetDatabase": "targetDatabaseJobLevel",
  "compareDDL": "true",
  "logLevel": "99",
  "tdTdSettings":
  {
    "forceUtility": "DSA",
    "dataStreams": 5,
    "sourceSessions": 10,
    "targetSessions": 10,
    "onlineArchive": "false"
  }
},
"jobSecurity":
{
  "ownerName": "user10",
  "readPermission": {
    "users" : [ "user1", "user2", "user3" ],
    "roles" : [ "role1", "role2", "role3" ]
  },
  "writePermission": {
    "users" : [ "user1", "user2", "user3" ],
    "roles" : [ ]
  },
  "executePermission": {
    "users" : [ "user1", "user3" ],
    "roles" : [ "role1" ]
  }
},
"objects":
{
  "database":
  [
    {
      "name": "sourceDb",

```



```

"compareDDL": "false",
"journaling": "false",
"replaceDatabase": "false",
"selection": "unselected",
"table":
[
  {
    "name": "srcTableA",
    "stagingDatabase": "stagingDBA",
    "stagingDatabaseForTable": "stagingDBAB",
    "targetDatabase": "targetDBA",
    "targetName": "tgtTableA",
    "validateRowCount": "ALL",
    "compareDDL": "false",
    "forceTargetStagingTable": "true",
    "selection": "included",
    "exportWithoutSpool": "false",
    "stagingToTarget": "DELETE_INSERT"
  },
  {
    "name": "srcTableB",
    "compareDDL": "true",
    "selection": "included",
    "exportWithoutSpool": "true",
    "partialTableParameters":
    {
      "sqlWhereClause": "where columnA < 100",
      "keyColumns":
      [
        "columnA"
      ]
    },
    "teradataParameters":
    {
      "copyStats": "false",
      "allowTPTLoadMultiset": "true",
      "journaling": "false",
      "overrideLockAccess": "false"
    }
  }
],
"trigger":
[

```

```

{
  "database": "databaseA",
  "subjectTableDatabase": "subjectDatabaseA",
  "table": "tableA",
  "name": "triggerA",
  "actionTime":
  {
    "action": "AFTER",
    "enabled": "NO"
  },
  "selection": "included"
},
{
  "database": "databaseB",
  "subjectTableDatabase": "subjectDatabaseB",
  "table": "tableB",
  "name": "triggerB",
  "actionTime":
  {
    "action": "AFTER",
    "enabled": "YES"
  },
  "selection": "included"
}
],
"view":
[
  {
    "name": "ViewA",
    "database": "ViewDatabase",
    "viewDataTable":
    {
      "targetTable": "tableA",
      "targetDatabase": "dbA"
    },
    "compareDDL": "false",
    "forceTargetStagingTable": "true",
    "selection": "included",
    "copyData": "false"
  },
  {
    "name": "ViewB",
    "database": "ViewDatabaseB",
    "viewDataTable":

```

```

        {
            "targetTable": "tableA",
            "targetDatabase": "dbA"
        },
        "compareDDL": "false",
        "selection": "included",
        "copyData": "false"
    }
],
"foreignServer":
[
    {
        "name": "foreignServer1",
        "selection": "included"
    },
    {
        "name": "foreignServer2",
        "selection": "included"
    }
],
"functionAliases":
[
    {
        "name": "faObj",
        "database": "dmguest",
        "selection": "included"
    },
    {
        "name": "FAObjForForeignSer",
        "database": "myDB",
        "selection": "included"
    }
]
"indices":
[
    {
        "name": "OrdersHI",
        "indexDatabase": "indexDB",
        "indexType": "hash_index",
        "selection": "included"
    },
    {
        "name": "sji",
        "indexDatabase": "indexDB",

```

```

        "indexType": "join_index",as
        "copyStats": true,
        "selection": "included"
    }
],
"schema":
[
    {
        "name": "avroRecordSchema",
        "compareDDL": true,
        "selection": "included"
    }
],
"macro":
[
    {
        "name": "macroA",
        "database": "macroDB",
        "selection": "included"
    },
    {
        "name": "macroB",
        "database": "macroDB",
        "selection": "included"
    }
],
"storedProcedure":
[
    {
        "name": "storedProcedureA",
        "database": "storedProcedureDB",
        "selection": "included"
    },
    {
        "name": "storedProcedureB",
        "database": "storedProcedureDB",
        "selection": "included"
    }
]
}

```

```
]
}
```

Related Information:

[Viewing Data Mover REST Samples](#)
[RESTful API Status Codes](#)
[RESTful API Security](#)
[RESTful API Object Types](#)

Encrypt Password (RESTful API)

Purpose

Create an encrypted password from a password entered as a parameter to use for Data Mover commands using the following URL and method:

Item	Description
URL	/datamover/encrypted_password
Method	POST

Request Parameters**plain**

Description: The type of text to encrypt
 JSON Data Type: String
 Required: Yes

Response Parameters**encrypted**

Description: Encrypted password
 JSON Data Type: String

Response Example

```
{
  "encrypted": "ENCRYPTEDPASSWORD"
}
```

Get Cloud Staging Area (RESTful API)

Overview

Retrieve the details of an existing cloud staging area using the following URL and method:

Item	Description
URL	/datamover/cloud-staging-areas/staging_area_name
Method	GET

Request Parameters

No request parameters required.

Response Parameters

name

Description: Name of the cloud staging area

JSON Data Type: String

Required: Yes

storage_type

Description: Specify cloud storage type. Currently only S3 is supported

JSON Data Type: String

Required: Yes

s3_properties

Description: AWS S3 information used to create the cloud staging area

JSON Data Type: Object ([S3Properties](#))

Required: No

source_target_pairs

Description: Source and target system information

JSON Data Type: JSON Array ([SourceTargetPair](#))

Required: Yes

messages

Description: Contains error and warning messages

JSON Data Type: Object ([Messages](#))

List Agents (RESTful API)

Purpose

List agents using the following URL and method:

Item	Description
URL	/datamover/agents
Method	GET

Request Parameters

No request parameters required.

Response Parameters

agentName

Description: Name that identifies the agent.

JSON Data Type: String

agentHostName

Description: Name of the host running the agent.

JSON Data Type: String

agentVersion

Description: Data Mover version for the agent.

JSON Data Type: Version

maxConcurrentTasks

Description: Maximum number of concurrent tasks the agent can run.

JSON Data Type: Integer

currentTaskCount

Descriptions: Number of tasks the agent is currently running.

JSON Data Type: Integer

major

Description: Number corresponding to the major version.

JSON Data Type: String

minor

Description: Number corresponding to the minor version.

JSON Data Type: String

maintenance

Description: Maintenance release number.

JSON Data Type: String

efix

Description: efix release number.

JSON Data Type: String

Response Examples

```
[
{
  "agentName" : "AgentBond",
  "agentHostName" : "dm-agent7",
  "agentVersion" : {
    "major" : "16",
    "minor" : "00",
    "maintenance" : "00",
    "efix" : "00"
  },
  "maxConcurrentTasks" : 5,
  "currentTaskCount" : 0
},
{
  "agentName" : "AgentKay",
  "agentHostName" : "dm-agent4",
  "agentVersion" : {
    "major" : "16",
    "minor" : "00",
    "maintenance" : "00",
    "efix" : "00"
  },
}
```



```

    "maxConcurrentTasks" : 5,
    "currentTaskCount" : 0
  }
]

```

This example shows an error message.

```

{
  "message": "Unable to communicate with DM Daemon. Make sure the broker url
and port are correct."
}

```

Related Information:

[Viewing Data Mover REST Samples](#)

[RESTful API Status Codes](#)

[RESTful API Security](#)

[RESTful API Object Types](#)

List Configuration (RESTful API)

Purpose

Display configuration settings for the daemon, stored procedures, table-driven interface of the daemon, and performance settings for the daemon.

There are two variations of the API that display information for the following:

- Specified property
- All properties specified in the request body

Item	Description
URL	/datamover/daemonProperties/ <i>propertyName</i> /datamover/daemonProperties
Method	GET

Request Parameters

No request parameters required.

Response Parameters

propertyName

Description: Configuration property name

JSON Data Type: String

Required: Yes

values

Description: Property values per system

JSON Data Type: JSON Array ([valueType](#))

Required: Yes

unit

Description: Unit of the value

JSON Data Type: String

Required: No

description

Description: Additional information

JSON Data Type: String

Required: Yes

systemPairs

Description: System pairs used to force job direction

JSON Data Type: JSON Array ([systemPairType](#))

Required: No

groupPools

Description: User group pools

JSON Data Type: JSON Array ([userGroupType](#))

Required: No

targetUserPools

Description: Systems for target user pool

JSON Data Type: JSON Array ([systemType](#))

Required: No

neverTargetSystems

Description: Systems never used as the target system

JSON Data Type: JSON Array (String)

Required: No

defaultDatabases

Description: Databases used as default target or staging databases at system level

JSON Data Type: JSON Array ([systemLevelDatabaseType](#))

Required: No

Response Examples

This example shows a specified property.

```
{
  "propertyName": "agentCollector.agentHeartbeatWaitMillis",
  "values":
  [
    {
      "value": "600000",
      "system": "ALL"
    }
  ],
  "description": "Purpose: To set the amount of time to wait for an
Agent heartbeat before assuming it has gone out of service in milliseconds.
Default: 600000"
}
```

This example shows all properties specified in response body.

Related Information:

[Viewing Data Mover REST Samples](#)

[RESTful API Status Codes](#)

[RESTful API Security](#)

[RESTful API Object Types](#)

List Cloud Staging Areas (RESTful API)

Overview

List all the existing cloud staging areas using the following URL and method:

Item	Description
URL	/datamover/cloud-staging-areas/
Method	GET

Request Parameters

No request parameters required.

Response Parameters

cloud_staging_areas

Description: Contains all current cloud staging areas

JSON Data Type: JSON Array ([CloudStagingArea](#))

Required: Yes

messages

Description: Contains error and warning messages

JSON Data Type: Object ([Messages](#))

Response Example

```
{
  "cloud_staging_areas": [
    {
      "name": "cs2-cloudstagingarea1",
      "storage_type": "s3",
      "source_target_pairs": [
        {
          "source_system": "sourceSystem1",
          "target_system": "targetSystem1"
        },
        {
          "source_system": "sourceSystem2",
          "target_system": "targetSystem2"
        }
      ]
    },
    {
      "name": "cs2-cloudstagingarea2",
      "storage_type": "s3",
      "source_target_pairs": [
        {
          "source_system": "sourceSystem3",
          "target_system": "targetSystem3"
        }
      ]
    }
  ]
}
```

```

    }
  ],
  "messages": {
    "errors": [],
    "warnings": []
  }
}

```

List Data Mover Information (RESTful API)

Purpose

Displays general information about the Data Mover service, such as a version number, using the following URL and method:

Item	Description
URL	/datamover/api
Method	GET

Response Parameters

VersionType

Description: Displays related Data Mover daemon information.

JSON data type: Object

Required: No

Request Parameters

The datamover_info RESTful API does not require specific response parameters.

Response Example

```

{
  "version":
  {
    "major": "16",
    "minor": "00",
    "maintenance": "00",
    "efix": "00"
  }
}

```

List Event Tables (RESTful API)

Overview

List the ID, host, and parent database of the event tables using the following URL and method:

Item	Description
URL	/datamover/event-tables
Method	GET

Request Parameters

No specific request parameters required.

Response Parameters

event_table_id

Description: Event table ID

JSON Data Type: String

event_table_host

Description: Name of the Teradata system where the event table is located

JSON Data Type: String

event_table_parent_database

Description: Name of the parent database for event table

JSON Data Type: String

host_user_name

Description: User name used to connect to the host where the event table is located

JSON Data Type: String

Response Examples

```
[
{
  "event_table_id" : "event1",
  "event_table_host" : "td1",
  "event_table_parent_database" : "parentdb1",
  "host_user_name" : "user1"
```

```

},
{
  "event_table_id" : "event2",
  "event_table_host" : "td2",
  "event_table_parent_database" : "parentdb1",
  "host_user_name" : "user2"
}
]

```

List Job Definition (RESTful API)

Purpose

Display the definition of a job using the following URL and method:

Item	Description
URL	<i>/datamover/jobs/job-name</i>
Method	GET

Request Parameters

No required request parameters.

Response Parameters

jobName

Description: Job name

JSON Data Type: String

uowid

Description: Unit of work id

JSON Data Type: String

sourceLogin

Description: Source system properties

JSON Data Type: Object ([LoginType](#))

targetLogin

Description: Target system properties

JSON Data Type: Object ([LoginType](#))

settings

Description: Job configuration properties

JSON Data Type: Object ([SettingsType](#))

objects

Description: Objects to be copied

JSON Data Type: Object ([DbObjectType](#))

Note:

jobSecurity details are not displayed in list job definition even though **jobSecurity** details are provided when creating or editing a job.

Response Example (Teradata to Teradata UserPool)

```
{
  "jobName": "tdToTdJob",
  "sourceLogin":
  {
    "teradata":
    {
      "tdpid": "sourceSystem",
      "username": "user",
      "password": "",
      "sessionCharset": "UTF8",
      "passwordEncrypted": "false"
    }
  },
  "targetLogin":
  {
    "userPool": "poolA"
  },
  "settings":
  {
    "priority": "MEDIUM",
    "overwriteExistingObjects": "true",
    "freezeJobSteps": "false",
    "targetDatabase": "targetDatabaseJobLevel",
    "compareDDL": "true",
    "logLevel": "99",
    "tdTdSettings":
    {
```



```

    "forceUtility": "DSA",
    "dataStreams": 5,
    "sourceSessions": 10,
    "targetSessions": 10,
    "onlineArchive": "false"
  }
},
"objects":
{
  "database":
  [
    {
      "name": "sourceDb",
      "compareDDL": "false",
      "journaling": "false",
      "replaceDatabase": "false",
      "selection": "unselected",
      "table":
      [
        {
          "name": "srcTableA",
          "stagingDatabase": "stagingDBA",
          "stagingDatabaseForTable": "stagingDBAB",
          "targetDatabase": "targetDBA",
          "targetName": "tgtTableA",
          "validateRowCount": "ALL",
          "compareDDL": "false",
          "forceTargetStagingTable": "true",
          "selection": "included",
          "exportWithoutSpool": "false",
          "stagingToTarget": "DELETE_INSERT"
        },
        {
          "name": "srcTableB",
          "compareDDL": "true",
          "selection": "included",
          "exportWithoutSpool": "true",
          "partialTableParameters":
          {
            "sqlWhereClause": "where columnA < 100",
            "keyColumns":
            [
              "columnA"
            ]
          }
        }
      ]
    }
  ]
}

```

```

        },
        "teradataParameters":
        {
            "copyStats": "false",
            "allowTPTLoadMultiset": "true",
            "journaling": "false",
            "overrideLockAccess": "false"
        }
    }
]
},
"trigger":
[
    {
        "database": "databaseA",
        "subjectTableDatabase": "subjectDatabaseA",
        "table": "tableA",
        "name": "triggerA",
        "actionTime":
        {
            "action": "AFTER",
            "enabled": "NO"
        },
        "selection": "included"
    },
    {
        "database": "databaseB",
        "subjectTableDatabase": "subjectDatabaseB",
        "table": "tableB",
        "name": "triggerB",
        "actionTime":
        {
            "action": "AFTER",
            "enabled": "YES"
        },
        "selection": "included"
    }
],
"view":
[
    {
        "name": "ViewA",
        "database": "ViewDatabase",

```

```

        "viewDataTable":
        {
            "targetTable": "tableA",
            "targetDatabase": "dbA"
        },
        "compareDDL": "false",
        "selection": "included",
        "copyData": "false"
    },
    {
        "name": "ViewB",
        "database": "ViewDatabaseB",
        "viewDataTable":
        {
            "targetTable": "tableA",
            "targetDatabase": "dbA"
        },
        "compareDDL": "false",
        "forceTargetStagingTable": "true",
        "selection": "included",
        "copyData": "false"
    }
],
"foreignServer":
[
    {
        "name": "foreignServer1",
        "selection": "included"
    },
    {
        "name": "foreignServer2",
        "selection": "included"
    }
],
"functionAliases":
[
    {
        "name": "faObj",
        "database": "dmguest",
        "selection": "included"
    },
    {
        "name": "FAObjForForeignSer",
        "database": "myDB",

```

```

        "selection": "included"
    }
]
"journal":
[
    {
        "name": "journal",
        "database": "journalDB",
        "selection": "included"
    }
],
"macro":
[
    {
        "name": "macroA",
        "database": "macroDB",
        "selection": "included"
    },
    {
        "name": "macroB",
        "database": "macroDB",
        "selection": "included"
    }
],
"indices":
[
    {
        "name": "OrdersHI",
        "indexDatabase": "indexDB",
        "indexType": "hash_index",
        "selection": "included"
    },
    {
        "name": "sjj",
        "indexDatabase": "indexDB",
        "indexType": "join_index",
        "copyStats": true,
        "selection": "included"
    }
],
"schema":
[
    {
        "name": "avroRecordSchema",

```

```

        "compareDDL": true,
        "selection": "included"
    },
],
"storedProcedure":
[
    {
        "name": "storedProcedureA",
        "database": "storedProcedureDB",
        "selection": "included"
    },
    {
        "name": "storedProcedureB",
        "database": "storedProcedureDB",
        "selection": "included"
    }
]
}

```

Related Information:
[Viewing Data Mover REST Samples](#)
[RESTful API Status Codes](#)
[RESTful API Security](#)
[RESTful API Object Types](#)

List Job Steps (RESTful API)

Purpose

Display the list of steps for a specified job without starting or running the job using the following URL and method:

Item	Description
URL	/datamover/jobs/ <i>job-name</i> /steps
Method	GET

Request Parameters

No required request parameters.

Response Parameters

jobName

Description: Name of the job

JSON Data Type: String

sourceLogin

Description: Properties to specify source system

JSON Data Type: Object ([LoginType](#))

targetLogin

Description: Properties to specify target system

JSON Data Type: Object ([LoginType](#))

steps

Description: Steps of the job

JSON Data Type: JSON Array ([RestStepListType](#))

sourceSessions

Description: Source session used

JSON Data Type: [SessionsAndStreamsType](#)

targetSessions

Description: Target session used

JSON Data Type: [SessionsAndStreamsType](#)

dataStreams

Description: Data Stream used

JSON Data Type: [SessionsAndStreamsType](#)

Example

```
{
  "jobName" : "job1",
  "sourceLogin" : {
    "teradata" : {
      "tdpid" : "source1",
      "username" : "sourceuser",
      "passwordEncrypted" : false
    }
  },
}
```

```

"targetLogin" : {
  "teradata" : {
    "tdpid" : "starget1",
    "username" : "targetuser",
    "passwordEncrypted" : false
  }
},
"steps" : [ {
  "type" : "VERIFY_JOB_OBJECTS",
  "id" : 2564
}, {
  "type" : "COMPARE_DDL",
  "id" : 2568
}, {
  "tasks" : [ {
    "type" : "DSATaskType",
    "dsaJobName" : "849_table",
    "jobModelJson" : "{\n  \"jobName\" : \"849_table\",\n
\\\"jobDescription\" : \"DSA job for data mover job: job1\",\n  \"jobType\" : 
\\\"DATA_MOVEMENT\",\n  \"dataDictionaryType\" : \"DATA\",\n  \"objectList\" : 
[ {\n    \"objectName\" : \"table1\",\n    \"objectType\" : \"TABLE\",\n
\\\"parentType\" : \"DATABASE\",\n    \"parentName\" : \"parent\"\n  } ],\n
\\\"jobSettings\" : {\n    \"skipStats\" : true,\n    \"online\" : 
false,\n    \"loggingLevel\" : \"Debug\",\n    \"blockLevelCompression\" : 
\\\"DEFAULT\",\n    \"runAsCopy\" : true,\n    \"queryband\" : 
\\\"SET QUERY_BAND='ApplicationName=DM;Version=16.20.50.01;'\",\n
\\\"numberParallelBuilds\" : 5,\n    \"nosync\" : false,\n
\\\"temperatureOverride\" : \"DEFAULT\",\n    \"disableFallback\" : false,\n
\\\"nowait\" : true,\n    \"streamsSoftlimit\" : 2,\n    \"skipJoinhashIndex\" : 
true\n  }\n}",
    "id" : 2569
  } ],
  "type" : "MOVE_TABLE_DATA",
  "id" : 2569
} ],
"sourceSessions" : {
  "value" : 2,
  "type" : "SYSTEM_GENERATED"
},
"targetSessions" : {
  "value" : 2,
  "type" : "SYSTEM_GENERATED"
},
"dataStreams" : {

```

```

    "value" : 2,
    "type" : "SYSTEM_GENERATED"
  }
}

```

List Jobs (RESTful API)

Purpose

Data Mover provides the following filter variations when listing jobs:

- Jobs that meet the specified conditions, including start time, end time, and status
- Jobs that ran and met the specified conditions, including start time, end time, status, and priority
- Jobs currently running that meet the specified conditions, including start time, status, and priority
- Jobs that have frozen steps

Item	Description
URL	/datamover/jobs /datamover/executedJobs /datamover/executingJobs /datamover/jobs?freezeJobStepsOnly= <i>value</i>
Method	GET

Request Parameters (/datamover/jobs)

This variation returns a list of job names that meet the specified conditions, including start time, end time, and status. If no parameters are supplied, a list of new and completed jobs is returned.

startTimeAfter

Description: Jobs started later than specified time

Restriction: Cannot be used with endTimeAfter

URL: /datamover/jobs?startTimeAfter=*value*

JSON Data Type: String (with ISO 8601 UTC date format), for example: 2015-07-01T17:38:12Z

Required: No

endTimeBefore

Description: Jobs finished earlier than specified time.

Restriction: Cannot be used with endTimeAfter

URL: /datamover/jobs?endTimeBefore=*value*

JSON Data Type: String (with ISO 8601 UTC date format), for example: 2015-07-01T17:38:12Z

Required: No

endTimeAfter

Description: Jobs finished later than specified time.

Restriction: Cannot be used with endTimeBefore or startTimeAfter.

URL: /datamover/jobs?endTimeAfter=*value*

JSON Data Type: String (with ISO 8601 UTC date format), for example: 2015-07-01T17:38:12Z

Required: No

freezeJobStepsOnly

Description: List only jobs that have freezeJobStepsOnly set to true

URL: /datamover/jobs?freezeJobStepsOnly=*value*

Accepted values: True or False

- true = list jobs that have freeze job steps
- false = return all jobs, regardless of freeze steps

JSON Data Type: Boolean

Required: No

jobName

Description: Job name

JSON Data Type: String

Required: No

latestJobsOnly

Description: List the most recent job that ran in the daemon.

URL: /datamover/jobs?latestJobsOnly=*value*

Accepted values: True or False

- true = list only the most recently executed job
- false = list all jobs

JSON Data Type: Boolean

Required: No

status

Description: Jobs with last starting status that matches one of the following:

- NEW: All new jobs
- INITIALIZING: All initializing jobs
- RUNNING: All running jobs
- BLOCKED: All blocked jobs
- COMPLETED: All successfully completed jobs
- FAILED: All failed jobs
- RESTARTING: All restarting jobs
- CANCELLED: All user-cancelled jobs
- QUEUED: All queued jobs

URL: `/datamover/jobs?status=value`

JSON Data Type: String

Required: No

Response Parameters (/datamover/jobs)**jobExecutionName**

Description: Started job name

JSON Data Type: String

jobName

Description: Job name (maximum characters: 237)

JSON Data Type: String

startTime

Description: Start time if job has started or null if job did not start

JSON Data Type: String (with ISO 8601 UTC date format), for example: 2015-07-01T17:38:12Z

endTime

Description: End time if job has started or null if job did not start

JSON Data Type: String (with ISO 8601 UTC date format), for example: 2015-07-01T17:38:12Z

status

Description: Status of the job (maximum characters: 12), where possible values are specified in the Request Parameters section

JSON Data Type: String

Priority

Description: Priority of the job if started (maximum characters: 11), where the value is one of the following:

- UNSPECIFIED
- LOW
- MEDIUM
- HIGH

JSON Data Type: String

Response Example (/datamover/jobs)

This example shows an array of jobs with a start time later than the specified time, no jobs are found, and an empty array.

```
[
  {
    "jobName" : "runningJob" ,
    "startTime" : "2015-07-16T17:14:13Z",
    "endTime" : null ,
    "status" : "RUNNING" ,
    "priority" : "MEDIUM" ,
    "links" :
    [
      {
        "rel" : "status" ,
        "href" : "http://server/datamover/
executingJobs/runningJob-1417744247669"

      },
      {
        "rel" : "past_executions" ,
        "href" : "http://server/datamover/
executedJobs?jobName=runningJob&startTimeAfter=<value>"

      }
    ]
  }
]
```

```

        "rel" : "job_steps" ,
        "href" : "http://server/datamover/jobs/runningJob/steps"

    }
]
},
{
    "jobName" : "successfulJob" ,
    "startTime" : "2015-07-16T17:14:13Z",
    "endTime" : "2015-07-16T19:14:13Z",
    "status" : "COMPLETED" ,
    "priority" : "MEDIUM" ,
    "links" :
    [
        {
            "rel" : "status" ,
            "href" : "http://server/datamover/
executedJobs/successfulJob-20141204175020"

        },
        {
            "rel" : "past_executions" ,
            "href" : "http://server/datamover/
executedJobs?jobName=successfulJob
                        &startTimeAfter=<value>
                        "

        }
    ]
},
{
    "jobName" : "failedJob" ,
    "startTime" : 1417744247669,
    "endTime" : 1417744323437,
    "status" : "FAILED" ,
    "priority" : "MEDIUM" ,
    "links" :
    [
        {

```

```

        "rel" : "status" ,
        "href" : "http://server/datamover/
executedJobs/failedJob-20141204175020"

    },
    {
        "rel" : "past_executions" ,
        "href" : "http://server/datamover/
executedJobs?jobName=failedJob
                        &startTimeAfter=<value>
                        "

    }
    {
        "rel" : "job_steps" ,
        "href" : "http://server/datamover/jobs/failedJob/steps"

    }
    ]
}
]

```

Request Parameters (/datamover/executedJobs)

This variation returns a list of job names that meet the specified conditions, including jobs that are new or are running.

startTimeAfter

Description: Jobs started later than specified time

Restriction: Cannot be used with endTimeAfter

URL: /datamover/executedJobs

JSON Data Type: String (with ISO 8601 UTC date format), for example: 2015-07-01T17:38:12Z

Required: No

endTimeBefore

Description: Jobs finished earlier than specified time

Restriction: Cannot be used with endTimeAfter

URL: /datamover/executedJobs?endTimeBefore=*value*

JSON Data Type: String (with ISO 8601 UTC date format), for example: 2015-07-01T17:38:12Z

Required: No

endTimeAfter

Description: Jobs finished later than specified time

Restriction: Cannot be used with `startTimeAfter` or `endTimeBefore`

URL: `/datamover/executedJobs?endTimeBefore=value`

JSON Data Type: String (with ISO 8601 UTC date format), for example: `2015-07-01T17:38:12Z`

Required: No

status

Description: Jobs with last run status that matches one of the following:

- COMPLETED: All successfully completed jobs
- FAILED: All failed jobs
- CANCELLED: All user-cancelled jobs

URL: `/datamover/executedJobs?endTimeBefore=value`

JSON Data Type: String

Required: No

jobName

Description: Started jobs mapped to the specified name

URL: `/datamover/executedJobs?jobName=value`

JSON Data Type: String

Required: No

Response Parameters (/datamover/executedJobs)

endTime

Description: End time if job has started or null if not started

JSON Data Type: String (with ISO 8601 UTC date format), for example: `2015-07-01T17:38:12Z`

estimatedJobSize

Description: High estimate of job size

JSON Data Type: Long

Output Level: 1

jobExecutionName

Description: Started job name

JSON Data Type: String

jobName

Description: Job name (maximum characters: 237)

JSON Data Type: String

startTime

Description: Start time if job has started or null if not started

JSON Data Type: String (with ISO 8601 UTC date format), for example: 2015-07-01T17:38:12Z

status

Description: Job status (maximum characters: 11), where possible values are listed in the Request Parameters section

JSON Data Type: String (with ISO 8601 UTC date format), for example: 2015-07-01T17:38:12Z

Priority

Description: Priority of the job if and when started (maximum characters: 11), where the value is one of the following:

- UNSPECIFIED
- LOW
- MEDIUM
- HIGH

JSON Data Type: String

submitTime

Description: Time when job was submitted

JSON Data Type: String

Output Level: 1

RESTful API Example (/datamover/executedJobs)

This example shows a started job, no jobs found, and an empty array.

```
[
  {
    "jobExecutionName" : "successfulJob-20141204175020" ,
    "jobName" : "successfulJob" ,
    "startTime" : "2015-07-01T17:38:12Z",
    "endTime" : "2015-07-01T19:38:12Z",
    "status" : "COMPLETED" ,
    "priority" : "MEDIUM" ,
    "links" :
    [
      {
        "rel" : "status" ,
        "href" : "http://server/datamover/
executedJobs/successfulJob-20141204175020"

      }
      {
        "rel" : "job_steps" ,
        "href" : "http://server/datamover/jobs/successfulJob/steps"

      }
    ]
  },
  {
    "jobExecutionName" : "successfulJob-20141204175111" ,
    "jobName" : "successfulJob" ,
    "startTime" : "2015-07-02T17:38:12Z",
    "endTime" : "2015-07-02T19:38:12Z",
    "status" : "COMPLETED" ,
    "priority" : "MEDIUM" ,
    "links" :
    [
      {
        "rel" : "status" ,
        "href" : "http://server/datamover/
executedJobs/successfulJob-20141204175111"

      }
      {
        "rel" : "job_steps" ,
```



```

        "href" : "http://server/datamover/jobs/successfulJob/steps"
    }
]
},
{
    "jobExecutionName" : "failedJob-20141204175020" ,
    "jobName" : "failedJob" ,
    "startTime" : "2015-07-03T17:38:12Z",
    "endTime" : "2015-07-03T19:38:12Z",
    "status" : "FAILED" ,
    "priority" : "MEDIUM" ,
    "links" :
    [
        {
            "rel" : "status" ,
            "href" : "http://server/datamover/
executedJobs/failedJob-20141204175020"

        }
        {
            "rel" : "job_steps" ,
            "href" : "http://server/datamover/jobs/failedJob/steps"

        }
    ]
}
]

```

Request Parameters (/datamover/executingJobs)

This variation returns all jobs currently running on the Data Mover daemon and started later than the specified time, where the value is one of the following:

- INITIALIZING
- RUNNING
- BLOCKED
- RESTARTING
- QUEUED

You can provide the following optional parameters:

startTimeAfter

Description: Jobs started later than specified time

Restriction: Cannot be used with endTimeAfter

URL: /datamover/executingJobs

JSON Data Type: String (with ISO 8601 UTC date format), for example: 2015-07-01T17:38:12Z

Required: No

status

Description: Jobs with last run status that matches one of the following:

- COMPLETED: All successfully completed jobs
- FAILED: All failed jobs
- CANCELLED: All user-cancelled jobs

URL: /datamover/executingJobs?status=*value*

JSON Data Type: String

Required: No

Response Parameters (/datamover/executingJobs)

jobExecutionName

Description: Started job name

JSON Data Type: String

jobName

Description: Job name (maximum characters: 237)

JSON Data Type: String

startTime

Description: Start time if job has been started or null if not started

JSON Data Type: String (with ISO 8601 UTC date format), for example: 2015-07-01T17:38:12Z

status

Description: Status of the job (maximum characters: 11), where possible values are specified in Request Parameters list

JSON Data Type: String

Priority

Description: Priority of the job if and when started (maximum characters: 11), where the value is one of the following:

- UNSPECIFIED
- LOW
- MEDIUM
- HIGH

JSON Data Type: String

RESTful API Example (/datamover/executingJobs)

This example shows a started job, no jobs found, and an empty array.

```
[
  {
    "jobExecutionName" : "runningJob-1417744247669" ,
    "jobName" : "runningJob" ,
    "startTime" : "2015-07-16T17:14:13Z",
    "status" : "RUNNING" ,
    "priority" : "MEDIUM" ,
    "links" :
    [
      {
        "rel" : "status" ,
        "href" : "http://server/datamover/
runningJobs/runningJob-1417744247669"

      },
      {
        "rel" : "history" ,
        "href" : "http://server/datamover/
executedJobs?jobName=runningJob&startTimeAfter=<value>"

      }
    ]
  },
  {
    "rel" : "job_steps" ,
    "href" : "http://server/datamover/jobs/runningJob/steps"

  }
]
]
```

Request Parameters (/datamover/jobs?freezeJobStepsOnly=value)**freezeJobStepsOnly**

Description: Jobs that have freeze job steps set to true

JSON Data Type: Boolean

Required: No

Accepted values: True or False

- true = list jobs that have freeze job steps
- false = return all jobs, regardless of freeze steps

Response Parameters (/datamover/jobs?freezeJobStepsOnly=value)**jobExecutionName**

Description: Started job name

JSON Data Type: String

jobName

Description: Job name (maximum characters: 237)

JSON Data Type: String

links

Description: List of links to job status, past job executions, and job steps

JSON Data Type: JSON Array ([LinkType](#))

Priority

Description: Priority of the job if and when started (maximum characters: 11), where the value is one of the following:

- UNSPECIFIED
- LOW
- MEDIUM
- HIGH

JSON Data Type: String

startTime

Description: Job start time

JSON Data Type: String (with ISO 8601 UTC date format), for example: 2015-07-01T17:38:12Z

status

Description: Status of job (maximum characters: 12), where possible values are specified in the Request Parameters description

JSON Data Type: String

Related Information:

[Viewing Data Mover REST Samples](#)

[RESTful API Status Codes](#)

[RESTful API Security](#)

[RESTful API Object Types](#)

List Policies (RESTful API)

Overview

List the policies on jobs, daemon advanced settings, and daemon access using the following URL and method:

Item	Description
URL	/datamover/policies
Method	GET

Note:

When security is enabled, only a Viewpoint or command line admin can retrieve daemon advanced and daemon access policy types. Non-admin users can only check policies on a specific job.

Request Header

Authorization

Description: Basic header authentication

Note:

User must be **dmcl_admin** if call is from Viewpoint.

JSON Data Type: String

Required: No, unless **securityMgmt** is enabled

Portlet-User

Description: Viewpoint user login

JSON Data Type: String

Required: No, unless call is made from Viewpoint

Portlet-Roles

Description: Roles associated with Viewpoint user login

JSON Data Type: String, separated by commas when more than one role exists

Required: No, unless call is made from Viewpoint

Request Parameters**resource_type**

Description: The type of resource associated with the policy. Valid values are:

- tdrn:datamover:daemon_access
- tdrn:datamover:daemon_advanced
- tdrn:datamover:job

JSON Data Type: String

Required: Yes

resource_name

Description: The resource name associated with the policy.

JSON Data Type: String

Required: No, unless **resource_type** is tdrn:datamover:job

Response Parameters

No response parameters required.

Examples

The following is a request example to list policies for daemon access:

```
/datamover/policies?resource_type=tdrn:datamover:daemon_access
```

The following example is of a successful response for daemon access policies:

```
[ { "service" : "datamover", "type" : "user", "principals" :
[ "dn186008" ], "actions" : [ "read", "write", "execute" ],
```

```

"resources" : [ "tdrn:datamover:daemon_access:*" ] },
{ "service" : "datamover", "type" : "role", "principals" :
[ "Administrator" ], "actions" : [ "read", "write", "execute" ],
"resources" : [ "tdrn:datamover:daemon_access:*" ] },
{ "service" : "datamover", "type" : "user", "principals" :
[ "tester_002" ], "actions" : [ "read", "write" ],
"resources" : [ "tdrn:datamover:daemon_access:*" ] },
{ "service" : "datamover", "type" : "role", "principals" : [ "User" ],
"actions" : [ "read" ], "resources" : [ "tdrn:datamover:daemon_access:*" ] },
{ "service" : "datamover", "type" : "user", "principals" :
[ "abcd" ], "actions" : [ "read", "write", "execute" ],
"resources" : [ "tdrn:datamover:daemon_access:*" ] },
{ "service" : "datamover", "type" : "user", "principals" :
[ "admin" ], "actions" : [ "read", "write", "execute" ],
"resources" : [ "tdrn:datamover:daemon_access:*" ] },
{ "service" : "datamover", "type" : "user", "principals" : [ "tester_001" ],
"actions" : [ "read" ], "resources" : [ "tdrn:datamover:daemon_access:*" ] },
{ "service" : "datamover", "type" : "user", "principals" : [ "test_002" ],
"actions" : [ "write" ], "resources" : [ "tdrn:datamover:daemon_access:*" ] } ]

```

The following example is of a failed response code when the user does not have permissions to retrieve policies:

```

{ "message" : "When security is on, only commandline super user or viewpoint
could retrieve policies for resource type.tdrn:datamover:daemon_access.The user
does not have the permission to run GET_GLOBAL_ACCESS_PERMISSIONS command" }

```

List Tasks (RESTful API)

Purpose

The `list_tasks` RESTful API displays a list of queued and running tasks, including information about status and execution.

The `list_tasks` RESTful API uses the following URL and method:

Item	Description
URL	/datamover/tasks
Method	GET

Request Parameters

jobName

Description: List tasks associated with specified job name

Required: No

taskId

Description: List tasks associated with specified task ID

Required: No

agentName

Description: List tasks running on specified agent

Required: No

taskStatusMode

Description: List tasks in one of the following specified modes:

- A (ALL): All modes
- Q (QUEUED): All queued tasks
- R (RUNNING) All running tasks

Required: No

Note:

If you do not specify a mode, Data Mover includes all modes.

Response Parameters

agentNames

Description: Agent currently running the task, or '-' if queued

JSON Data Type: String Array

jobName

Description: Name of the associated job

JSON Data Type: String

jobPriority

Description: Priority of the associated job, where possible value is one of the following:

- UNSPECIFIED
- LOW
- MEDIUM
- HIGH

JSON Data Type: String

lastUpdateTime

Description: Time of last update.

JSON Data Type: Long

queueName

Description: Queue in which the task is waiting, or NOT_QUEUED if running

JSON Data Type: String

queueOrder

Description: Task position in the queue, or '-' if running

JSON Data Type: String

srcSystem

Description: Database source system from which the task is moving data

JSON Data Type: String

targetSystem

Description: Database system to which task is moving data

JSON Data Type: String

taskId

Description: Task ID

JSON Data Type: Integer

taskStatus

Description: Status of the task, where the value is one of the following:

- QUEUED
- RUNNING

JSON Data Type: String

targetUserPoolName

Description: Pool name, or '-' if none used

JSON Data Type: String

utility

Description: Utility on which task depends

JSON Data Type: String

Response Example

```
[
  {
    "taskID": "499",
    "jobName": "ast-1",
    "taskStatus": "QUEUED",
    "agentNames": [ ],
    "queueName": "LOAD_SLOT",
    "utility": "TPTAPI_LOAD",
    "srcSystem" : "10.25.46.85",
    "targetSystem": "sdt16849.labs.teradata.com"
    "jobPriority" : "LOW",
    "lastUpdateTime" : 1536693540772
  }
]
```

Move Job (RESTful API)

Overview

The move RESTful API moves the specified job using the following URL and method:

Item	Description
URL	/datamover/executingJobs
Method	POST

Request Parameters**jobName**

Description: Job name

JSON Data Type: String

Required: Yes

sync

Description: Sync use specification

Note:

If `sync` is `false` (default), the start command returns status when the job execution name is generated. If `sync` is `true`, the start command returns status when the job is complete.

JSON Data Type: String

Required: Boolean

Required: No

uowid

Description: Unit of work ID

JSON Data Type: String

Required: No

saveChanges

Description: Update original job definition with specified parameters

JSON Data Type: Boolean

Required: No

sourceLogin

Description: Source system properties

JSON Data Type: Object ([LoginType](#))

Required: No

targetLogin

Description: Target system properties

JSON Data Type: Object ([LoginType](#))

Required: No

settings

Description: Job configuration properties

JSON Data Type: Object ([SettingsType](#))

Required: No

jobSecurity

Description: Job security details

JSON Data Type: Object ([JobSecurityType](#))

Required: No

objects

Description: Objects to be copied

JSON Data Type: Object ([DbObjectType](#))

Required: No

Request Examples

```
{
  "jobName": "tdToTdJob",
  "sourceLogin":
  {
    "teradata":
    {
      "tdpid": "sourceSystem",
      "username": "user",
      "password": "password",
      "sessionCharset": "UTF8",
      "passwordEncrypted": "false"
    }
  },
  "targetLogin":
  {
    "userPool": "poolA"
  },
  "settings":
  {
    "priority": "MEDIUM",
    "overwriteExistingObjects": "true",
    "freezeJobSteps": "false",
    "targetDatabase": "targetDatabaseJobLevel",
    "compareDDL": "true",
    "logLevel": "99",
    "tdTdSettings":
    {
      "forceUtility": "DSA",
      "dataStreams": 5,

```

```

        "sourceSessions": 10,
        "targetSessions": 10,
        "onlineArchive": "false"
    }
},
"jobSecurity":
{
    "ownerName": "user10",
    "readPermission": {
        "users" : [ "user1", "user2", "user3" ],
        "roles" : [ "role1", "role2", "role3" ]
    },
    "writePermission": {
        "users" : [ "user1", "user2", "user3" ],
        "roles" : [ ]
    },
    "executePermission": {
        "users" : [ "user1", "user3" ],
        "roles" : [ "role1" ]
    }
},
"objects":
{
    "database":
    [
        {
            "name": "sourceDb",
            "compareDDL": "false",
            "journaling": "false",
            "replaceDatabase": "false",
            "selection": "unselected",
            "table":
            [
                {
                    "name": "srcTableA",
                    "stagingDatabase": "stagingDBA",
                    "stagingDatabaseForTable": "stagingDBAB",
                    "targetDatabase": "targetDBA",
                    "targetName": "tgtTableA",
                    "validateRowCount": "ALL",
                    "compareDDL": "false",
                    "forceTargetStagingTable": "true",
                    "selection": "included",

```

```

        "exportWithoutSpool": "false",
        "stagingToTarget": "DELETE_INSERT"
    },
    {
        "name": "srcTableB",
        "compareDDL": "true",
        "selection": "included",
        "exportWithoutSpool": "true",
        "partialTableParameters": {
            "sqlWhereClause": "where columnA < 100",
            "keyColumns": [
                "columnA"
            ]
        },
        "teradataParameters": {
            "copyStats": "false",
            "allowTPTLoadMultiset": "true",
            "journaling": "false",
            "overrideLockAccess": "false"
        }
    }
]
},
"trigger": [
    {
        "database": "databaseA",
        "subjectTableDatabase": "subjectDatabaseA",
        "table": "tableA",
        "name": "triggerA",
        "actionTime": {
            "action": "AFTER",
            "enabled": "NO"
        },
        "selection": "included"
    },
    {
        "database": "databaseB",
        "subjectTableDatabase": "subjectDatabaseB",

```

```

        "table": "tableB",
        "name": "triggerB",
        "actionTime":
        {
            "action": "AFTER",
            "enabled": "YES"
        },
        "selection": "included"
    }
],
"view":
[
    {
        "name": "ViewA",
        "database": "ViewDatabase",
        "viewDataTable":
        {
            "targetTable": "tableA",
            "targetDatabase": "dbA"
        },
        "compareDDL": "false",
        "selection": "included",
        "copyData": "false"
    },
    {
        "name": "ViewB",
        "database": "ViewDatabaseB",
        "viewDataTable":
        {
            "targetTable": "tableA",
            "targetDatabase": "dbA"
        },
        "compareDDL": "false",
        "forceTargetStagingTable": "true",
        "selection": "included",
        "copyData": "false"
    }
],
"foreignServer":
[
    {
        "name": "foreignServer1",
        "selection": "included"
    },

```

```

        {
            "name": "foreignServer2",
            "selection": "included"
        }
    ],
    "functionAliases":
    [
        {
            "name": "faObj",
            "database": "dmguest",
            "selection": "included"
        },
        {
            "name": "FAObjForForeignSer",
            "database": "myDb",
            "selection": "included"
        }
    ]
    "indices":
    [
        {
            "name": "OrdersHI",
            "indexDatabase": "indexDB",
            "indexType": "hash_index",
            "selection": "included"
        },
        {
            "name": "sji",
            "indexDatabase": "indexDB",
            "indexType": "join_index",as
            "copyStats": true,
            "selection": "included"
        }
    ],
    "schema":
    [
        {
            "name": "avroRecordSchema",
            "compareDDL": true,
            "selection": "included"
        }
    ],
    "journal":
    [

```



```

        {
            "name": "journal",
            "database": "journalDB",
            "selection": "included"
        }
    ],
    "macro":
    [
        {
            "name": "macroA",
            "database": "macroDB",
            "selection": "included"
        },
        {
            "name": "macroB",
            "database": "macroDB",
            "selection": "included"
        }
    ],
    "storedProcedure":
    [
        {
            "name": "storedProcedureA",
            "database": "storedProcedureDB",
            "selection": "included"
        },
        {
            "name": "storedProcedureB",
            "database": "storedProcedureDB",
            "selection": "included"
        }
    ]
}

```

```
]
}
```

Related Information:

[Viewing Data Mover REST Samples](#)
[RESTful API Status Codes](#)
[RESTful API Security](#)
[RESTful API Object Types](#)

Restart Job (RESTful API)

Purpose

The restart RESTful API restarts a job using the following URL and method:

Item	Description
URL	/datamover/executingJobs
Method	POST

Request Parameters**action**

Description: Value "restart" to distinguish from other POST commands on /
executingJobs (not case-sensitive)

JSON Data Type: String

Required: Yes

jobExecutionName

Description: Job execution name for restart

JSON Data Type: String

Required: Yes

uowid

Description: Unit of work ID

JSON Data Type: String

Required: No

sync

Description: Specifies if the command is sync

JSON Data Type: Boolean

Required: No

Response Parameters

jobExecutionName

Description: Job execution name if job executed, or blank if not executed

JSON Data Type: String

status

Description: Status of job

JSON Data Type: String

Response Examples

In this example, sync is false.

```
{
  "jobExecutionName" : "executedJob-20141204175020" ,
  "status":"started"
  "links" : [{
    "rel" : "status"
    "href" : "http://server/datamover/jobs/executedJob-20141204175020"
  },
  {
    "rel" : "stop"
    "href" : "http://server/datamover/jobs/executedJob-20141204175020"
  }
  ]
}
```

In this example, sync is true.

```
{
  "jobExecutionName" : "executedJob-20141204175020" ,
  "status" : "FAILED" ,
  "links" : [{
    "rel" : "status"
    "href" : "http://server/datamover/jobs/executedJob-20141204175020"
  }
  ]
}
```

```
]
}
```

In this example, the job restart failed.

```
{
  "error" : "The job failed to execute because another job has
already started"
}
```

restart RESTful API Example

```
{
  "action": "restart",
  "jobExecutionName": "job_restart-date-time",
  "sync": "true"
}
```

Related Information:

[Viewing Data Mover REST Samples](#)

[RESTful API Status Codes](#)

[RESTful API Security](#)

[RESTful API Object Types](#)

Restore Daemon (RESTful API)

Purpose

Restore the repository and configuration files of the Data Mover daemon after a failover, hardware change, or similar scenarios. Always check the restore files to verify the restore has completed successfully before resuming Data Mover operations.

Restore the repository and configuration files of the Data Mover daemon Item	Description
URL	/datamover/restores
Method	POST

Request Parameters

directory

Description: Directory where the backup is stored

JSON Data Type: String

Required: Yes

Response Parameters

message

Description: States if the restore was successful or failed

JSON Data Type: String

Response Example

```
{
  "message": "Daemon restore job is scheduled, restore files are retrieved
from : /var/opt/teradata/user1/daemon_bu/ . Daemon will be restarted"
}
```

Save Configuration (RESTful API)

Purpose

The save_configuration RESTful API updates configuration settings, performance settings, table-driven interface of the daemon, and configuration settings for stored procedures.

There are two variations of the save_configuration RESTful API that do the following:

- Update specified property
- Update all properties specified in the request body

The save_configuration RESTful API uses the following URL and method:

Item	Description
URL	/datamover/daemonProperties/ <i>propertyName</i> /datamover/daemonProperties
Method	PUT

Request Parameters (propertyName)

This variation of the save_configuration RESTful API updates the specified property.

propertyName

Description: Configuration property name

JSON Data Type: String

Required: Yes

values

Description: Property values per system

JSON Data Type: JSON Array ([valueType Object](#))

Required: Yes

unit

Description: Unit of the value

JSON Data Type: String

Required: No

description

Description: Additional information

JSON Data Type: String

Required: Yes

systemPairs

Description: System pairs used to force job direction

JSON Data Type: JSON Array ([systemPairType Object](#))

Required: No

groupPools

Description: User group pools

JSON Data Type: JSON Array ([userGroupType Object](#))

Required: No

targetUserPools

Description: Systems for target user pool

JSON Data Type: JSON Array ([systemType Object](#))

Required: No

neverTargetSystems

Description: Systems never used as the target system

JSON Data Type: JSON Array (String)

Required: No

defaultDatabases

Description: Databases used as default target or staging databases at system level

JSON Data Type: JSON Array ([systemLevelDatabaseType Object](#))

Required: No

Response Parameters**messages**

Description: Message indicating if the property value was update or remained the same.

JSON Data Type: JSON Array (String)

errors

Description: Error message if property was not saved.

JSON Data Type: JSON Array (String)

Response Example (propertyName)

```
{
  "error": "The maximum value for property blocked.job.maxAllowedLimit cannot
be greater that 25% of the maximum concurrent job limit. "
}
```

save_configuration RESTful API Example (propertyName)

```
{
  "propertyName": "agentCollector.agentHeartbeatWaitMillis",
  "values":
  [
    {
      "value": "600000",
      "system": "ALL"
    }
  ],
  "description": "Purpose: To set the amount of time to wait for an
Agent heartbeat before assuming it has gone out of service in milliseconds.
Default: 600000"
}
```

Request Parameters (Properties)

This variation of the `save_configuration` RESTful API updates all properties specified in the request body.

propertyName

Description: Configuration property name

JSON Data Type: String

Required: Yes

values

Description: Property values per system

JSON Data Type: JSON Array([valueType Object](#))

Required: Yes

unit

Description: Unit of the value

JSON Data Type: String

Required: No

description

Description: Additional information

JSON Data Type: String

Required: Yes

systemPairs

Description: System pairs used to force job direction

JSON Data Type: JSON Array([systemPairType Object](#))

Required: No

groupPools

Description: User group pools

JSON Data Type: JSON Array([userGroupType Object](#))

Required: No

targetUserPools

Description: Systems for target user pool

JSON Data Type: JSON Array([systemType Object](#))

Required: No

neverTargetSystems

Description: Systems never used as the target system

JSON Data Type: JSON Array(String)

Required: No

defaultDatabases

Description: Databases used as default target or staging databases at system level

JSON Data Type: JSON Array([systemLevelDatabaseType Object](#))

Required: No

save_configuration RESTful API Example (Properties)**Response Example (Properties)**

```
{
  "error": "The maximum value for property blocked.job.maxAllowedLimit cannot
be greater than 25% of the maximum concurrent job limit. "
}
```

Related Information:

[Viewing Data Mover REST Samples](#)

[RESTful API Status Codes](#)

[RESTful API Security](#)

[RESTful API Object Types](#)

Start Job (RESTful API)**Overview**

The start RESTful API starts the specified job using the following URL and method:

Item	Description
URL	/datamover/executingJobs
Method	POST

Note:

The start command creates a job, even if the job does not already exist, when parameters in the `objects` field meet job specification requirements. When this occurs, the `saveChanges` parameters are ignored and a new job is saved.

Request Parameters**jobName**

Description: Job name

JSON Data Type: String

Required: Yes

sync

Description: Sync use specification

Note:

If `sync` is `false` (default), the start command returns status when the job execution name is generated. If `sync` is `true`, the start command returns status when the job is complete.

JSON Data Type: String

Required: Boolean

Required: No

uowid

Description: Unit of work ID

JSON Data Type: String

Required: No

saveChanges

Description: Update original job definition with specified parameters

JSON Data Type: Boolean

Required: No

sourceLogin

Description: Source system properties

JSON Data Type: Object ([LoginType](#))

Required: No

targetLogin

Description: Target system properties

JSON Data Type: Object ([LoginType](#))

Required: No

settings

Description: Job configuration properties

JSON Data Type: Object ([SettingsType](#))

Required: No

jobSecurity

Description: Job security details

JSON Data Type: Object ([JobSecurityType](#))

Required: No

objects

Description: Objects to be copied

JSON Data Type: Object ([DbObjectType](#))

Required: No

Response Parameters**jobExecutionName**

Description: Job execution name if job executed, or blank if not executed

JSON Data Type: String

status

Description: Status of job

JSON Data Type: String

Response Example

In this example, job start failed due to non-existent job and insufficient new job specifications.

```
{
  "message" : "Job not found and could not create a new job: Error: Cannot get
database version info for system dmsmp2"
}
```

start RESTful API Example

```
{
  "jobName" : "executedJob" ,
  "sync" : "true",
  "saveChanges" : "true",
  "sourceLogin":
  {
    "teradata":
    {
      "tdpid":"dmdev"
      "username": "dbc",
      "password": "dbc"
    }
  },
}
```

Related Information:

[Viewing Data Mover REST Samples](#)

[RESTful API Status Codes](#)

[RESTful API Security](#)

[RESTful API Object Types](#)

Status Job (RESTful API)

Overview

The status RESTful API displays the status of an executed job using the following URL and method:

Item	Description
URL	<i>/datamover/executedJobs/executed-job-name</i> <i>/datamover/executingJobs/executed-job-name</i>
Method	GET

Request Parameters

outputLevel

URL: /datamover/executingJobs/*executed-job-name*?outputLevel=*value*

URL: /datamover/executedJobs/*executed-job-name*?outputLevel=*value*

Description: Specifies output level of the job status. The output level options are defined in the following table:

Output Level	Description
1	Overall job status (default)
2	Status of each job step
3	Status of each job task
4	Detailed log information You must use outputLevel=4 when the job status is NEW to get detailed job information.

Note:

If the **outputLevel** parameter is not specified, then it defaults to output_level 1.

Required: No

Response Parameters

currentStep

Description: Most current step ID for the job, null if never executed

JSON Data Type: Integer

Output Level: 2

endTime

Description: End time of job, null if never executed

JSON Data Type: String

Output Level: 1

estimatedJobSize

Description: High estimate of job size

JSON Data Type: Long

Output Level: 1

jobExecutionId

Description: Job execution identification

JSON Data Type: Long

Output Level: 1

jobName

Description: Job name (maximum characters: 237)

JSON Data Type: String

Output Level: 1

jobExecutionName

Description: Job execution name

JSON Data Type: String

Output Level: 1

lifeCycle

Description: Lists all the events during job creation and execution

JSON Array: Object ([LifeCycleStatus](#))

Output Level: 3

log

Description: Displays the utility log output

JSON Data Type: String

Output Level: 4

rowCount

Description: Row count results of the source and target objects if row count validation steps exist

JSON Array: Object ([RowCountStatus](#))

Output Level: 3

startTime

Description: Start time of job, null if never executed

JSON Data Type: String

Output Level: 1

status

Description: Status of the job (maximum characters: 12), where possible values are listed in the **status** description for Request Parameters

JSON Data Type: String

Output Level: 1

steps

Description: Lists all the last steps of the job, null if never executed

JSON Array: Object ([StepType](#))

Output Level: 2

streams

Description: Lists streams, if available

JSON Array: Object ([StreamInfo](#))

Output Level: 3

submitTime

Description: Time when job was submitted

JSON Data Type: String

Output Level: 1

tasks

Description: Lists all the individual tasks for each object, null if never executed

JSON Array: Object ([TaskStatus](#))

Output Level: 3

Response Example

The following example shows an array of job executions with the most common status. The links generated depend on job status.

```
{
  "jobName" : "job1-20200128112954",
  "jobExecutionName" : "job1-20200128112954",
```

```

"jobExecutionId" : 25,
"currentStep" : 3,
"startTime" : 1580239796717,
"endTime" : 1580239811907,
"status" : "COMPLETED",
"steps" : [ {
  "id" : 1,
  "status" : "COMPLETED",
  "type" : "VERIFY_JOB_OBJECTS",
  "startTime" : 1580239799361,
  "endTime" : 1580239800076
}, {
  "id" : 2,
  "status" : "COMPLETED",
  "type" : "COMPARE_DDL",
  "startTime" : 1580239800186,
  "endTime" : 1580239802157
}, {
  "id" : 3,
  "status" : "COMPLETED",
  "type" : "MOVE_TABLE_DATA",
  "startTime" : 1580239802203,
  "endTime" : 1580239811792
} ],
"tasks" : [ {
  "id" : 2570,
  "movePhase" : "VALIDATING",
  "status" : "COMPLETE",
  "timestamp" : 1580239799774
}, {
  "id" : 2571,
  "movePhase" : "COMPARE_DDL",
  "status" : "COMPLETE",
  "agentName" : "Agent1",
  "utility" : "SQL",
  "timestamp" : 1580239801883
}, {
  "id" : 2572,
  "movePhase" : "MOVING_DATA",
  "status" : "COMPLETE",
  "objectType" : "table",
  "agentName" : "Agent1",
  "utility" : "DSA",
  "timestamp" : 1580239804392
} ]

```



```

    } ],
    "links" : [ {
      "rel" : "self",
      "href" : "https://localhost:1443/datamover/executedJobs/
job1-20200128112954?outputLevel=3"
    } ]

```

Related Information:
[Viewing Data Mover REST Samples](#)
[RESTful API Status Codes](#)
[RESTful API Security](#)
[RESTful API Object Types](#)

Stop Job (RESTful API)

The stop API stops a job using the following URL and method:

Item	Description
URL	/datamover/executingJobs/ <i>executed-job-name</i>
Method	DELETE

Request Parameters

The stop RESTful API does not require specific request parameters.

Response Parameters

The stop RESTful API does not include specific response parameters.

Response Example

```
{
  "message": "Error: job definition [job name] not found"
}
```

Related Information:

[Viewing Data Mover REST Samples](#)

[RESTful API Status Codes](#)

[RESTful API Security](#)

[RESTful API Object Types](#)

Update Job Priority (RESTful API)

Overview

Update the priority for running jobs using the following URL and method:

Item	Description
URL	/datamover/executingJobs/job-name/priority
Method	PUT

Request Parameters

job_priority

Description: New job priority value

JSON Data Type: String of LOW, MEDIUM, or HIGH

Required: Yes

Response Parameters

message

Description: Message explaining failure, no message for successful update

JSON Data Type: String

Response Examples

```
{
  "message" : "Must provide job_priority."
}
```

```
{
  "message" : "Can not find the job in the queue, not updating job priority"
}
```

Update Job Steps (RESTful API)

Overview

Update job steps for a specified job without starting or running the job using the following URL and method:

Item	Description
URL	/datamover/jobs/job-name/steps
Method	PUT

Request Parameters

No request parameters required.

Response Parameters

No response parameters required.

Response Example

```
{
  "error": "Job cannot be updated when the update command is being run with
the same job name."
}
```

Related Information:

[Viewing Data Mover REST Samples](#)

[RESTful API Status Codes](#)

[RESTful API Security](#)

[RESTful API Object Types](#)

RESTful API JSON Objects Structure Examples

Overview

With RESTful API, job details are specified in JSON format. This section provides examples of the JSON object structure for Teradata .

Teradata to Teradata REST Example

The following example shows the structure for creating a job, Teradata to Teradata:

```
{
  "jobName": "tdToTdJob",
  "sourceLogin":
  {
    "teradata":
    {
      "tdpid": "sourceSys",
    }
    "userPool": "poolA",
  },
  "targetLogin":
  {
    "teradata":
    {
      "tpid": "targetSys"
    },
    "userPool": "poolA"
  },
  "settings":
  {
    "priority": "MEDIUM",
    "overwriteExistingObjects": "true",
    "freezeJobSteps": "false",
    "targetDatabase": "targetDatabaseJobLevel",
    "compareDDL": "true",
    "logLevel": "99",
    "map" : "mapname1",
    "colocate" : "colocatename1",
    "tdTdSettings":
    {
      "forceUtility": "DSA",
      "dataStreams": 5,
      "sourceSessions": 10,
      "targetSessions": 10,
      "onlineArchive": "false"
    }
  },
  "jobSecurity":
  {
```

```

    "ownerName": "user10",
    "readPermission": {
      "users" : [ "user1", "user2", "user3" ],
      "roles" : [ "role1", "role2", "role3" ]
    },
    "writePermission": {
      "users" : [ "user1", "user2", "user3" ],
      "roles" : [ ]
    },
    "executePermission": {
      "users" : [ "user1", "user3" ],
      "roles" : [ "role1" ]
    }
  },
  "objects":
  {
    "database":
      [
        {
          "name": "sourceDb",
          "compareDDL": "false",
          "journaling": "false",
          "replaceDatabase": "false",
          "map" : "mapname1",
          "colocate" : "colocatename1",
          "selection": "unselected",
          "table":
            [
              {
                "name": "srcTableA",
                "stagingDatabase": "stagingDBA",
                "stagingDatabaseForTable": "stagingDBAB",
                "targetDatabase": "targetDBA",
                "targetName": "tgtTableA",
                "validateRowCount": "ALL",
                "compareDDL": "false",
                "forceTargetStagingTable": "true",
                "selection": "included",
                "exportWithoutSpool": "false",
                "map" : "mapname1",
                "colocate" : "colocatename1",
                "stagingToTarget": "DELETE_INSERT"
              },
              {

```

```

        "name": "srcTableB",
        "compareDDL": "true",
        "selection": "included",
        "exportWithoutSpool": "true",
        "map" : "mapname1",
        "colocate" : "colocatename1",
        "partialTableParameters":
        {
            "sqlWhereClause": "where columnA < 100",
            "keyColumns":
            [
                "columnA"
            ]
        },
        "teradataParameters":
        {
            "copyStats": "false",
            "allowTPTLoadMultiset": "true",
            "journaling": "false",
            "overrideLockAccess": "false"
        }
    }
]
},
"trigger":
[
    {
        "database": "databaseA",
        "subjectTableDatabase": "subjectDatabaseA",
        "table": "tableA",
        "name": "triggerA",
        "actionTime":
        {
            "action": "AFTER",
            "enabled": "NO"
        },
        "selection": "included"
    },
    {
        "database": "databaseB",
        "subjectTableDatabase": "subjectDatabaseB",
        "table": "tableB",
        "name": "triggerB",

```

```

        "actionTime":
        {
            "action": "AFTER",
            "enabled": "YES"
        },
        "selection": "included"
    }
],
"view":
[
    {
        "name": "ViewA",
        "database": "ViewDatabase",
        "viewDataTable":
        {
            "targetTable": "tableA",
            "targetDatabase": "dbA"
        },
        "compareDDL": "false",
        "selection": "included",
        "copyData": "false",
    },
    {
        "name": "ViewB",
        "database": "ViewDatabaseB",
        "viewDataTable":
        {
            "targetTable": "tableA",
            "targetDatabase": "dbA"
        },
        "compareDDL": "false",
        "forceTargetStagingTable": "true",
        "selection": "included",
        "copyData": "false",
    }
],
"foreignServer":
[
    {
        "name": "foreignServer1",
        "selection": "included",
        "map" : "mapname1",
        "colocate" : "colocatename1"
    }
]

```

```

    },
    {
        "name": "foreignServer2",
        "selection": "included",
        "map" : "mapname1",
        "colocate" : "colocatename1"
    }
],
"functionAliases":
[
    {
        "name": "faObj",
        "database": "dmguest",
        "selection": "included"
    },
    {
        "name": "FAObjForForeignSer",
        "database": "myDB",
        "selection": "included"
    }
]
"indices":
[
    {
        "name": "OrdersHI",
        "indexDatabase": "indexDB",
        "indexType": "hash_index",
        "map" : "mapname1",
        "colocate" : "colocatename1",
        "selection": "included"
    },
    {
        "name": "sjj",
        "indexDatabase": "indexDB",
        "indexType": "join_index",as
        "copyStats": true,
        "map" : "mapname1",
        "colocate" : "colocatename1",
        "selection": "included"
    }
],
"schema":
[
    {

```



```

        "name": "avroRecordSchema",
        "compareDDL": true,
        "selection": "included"
    },
],
"journal":
[
    {
        "name": "journal",
        "database": "journalDB",
        "selection": "included"
    }
],
"macro":
[
    {
        "name": "macroA",
        "database": "macroDB",
        "selection": "included"
    },
    {
        "name": "macroB",
        "database": "macroDB",
        "selection": "included"
    }
],
"storedProcedure":
[
    {
        "name": "storedProcedureA",
        "database": "storedProcedureDB",
        "selection": "included"
    },
    {
        "name": "storedProcedureB",
        "database": "storedProcedureDB",
        "selection": "included"
    }
]
}

```

The following example shows target and source logins using an encrypted password when creating a job, Teradata to Teradata:

```
{
  "jobName": "tdToTdJob",
  "sourceLogin":
  {
    "teradata":
    {
      "tdpid": "sourceSys",
      "username": "userA",

      "password": "76bd3f9e9c8a2a4e9f76ce57eac59ec97a33ef6157fc8aa730333513a53574c8",
      "sessionCharset": "UTF8",
      "passwordEncrypted": true
    }
  },
  "targetLogin":
  {
    "teradata":
    {
      "tdpid": "targetSys",
      "username": "userB",

      "password": "76bd3f9e9c8a2a4e9f76ce57eac59ec97a33ef6157fc8aa730333513a53574c8",
      "sessionCharset": "UTF8",
      "passwordEncrypted": true
    }
  },
  ...
}
```

The following example shows target and source logins using a plain text password when creating a job, Teradata to Teradata:

```
{
  "jobName": "tdToTdJob",
  "sourceLogin":
  {
    "teradata":
    {
      "tdpid": "sourceSys",
      "username": "userA",
      "password": "passwordA",
      "sessionCharset": "UTF8",
      "passwordEncrypted": false
    }
  },
  ...
}
```

```

    }
  },
  "targetLogin":
  {
    "teradata":
    {
      "tdpid": "targetSys",
      "username": "userB",
      "password": "passwordB",
      "sessionCharset": "UTF8",
      "passwordEncrypted": false
    }
  },
  ...

```

RESTful API Object Types

This section lists the object types shared by the Data Mover commands that support RESTful API.

ActionTimeType

Name	JSON Data Type	Required
enabled	String	Yes
action	String	Yes

BucketProperties

Name	Description	JSON Data Type
bucket_name	Name of the bucket in S3	String
prefix_list	List of prefixes within an S3 bucket	JSON Array (PrefixProperties)

BucketsByRegionProperties

Name	Description	JSON Data Type
region	AWS Region	String
buckets	Information for buckets in the specified region	JSON Array (BucketProperties)

ColumnType

Name	JSON Data Type	Required
name	String	No
targetName	String	No
type	String	No
targetType	String	No
size	String	No
charset	String	No
allowNull	String	No
allowDuplicate	String	No
isPrimaryIndex	String	No

CloudStagingArea

Name	Description	JSON Data Type
name	Name of the cloud staging area	String
storage_type	Cloud storage type of the area	String
source_target_pairs	Source and target systems for the area	JSON Array (SourceTargetPair)

CloudStagingAreaType

Name	Description	JSON Data Type	Required
Name Note: Required to run cloud staging copy jobs.	Name of the cloud staging area	String	No

DBObjectType

Name	JSON Data Type	Required
database	JSON Array (DbType)	No

Name	JSON Data Type	Required
trigger	JSON Array (TriggerType)	No
indices	JSON Array (IndexType)	No
view	JSON Array (ViewType)	No
foreignServer	JSON Array (ForeignServerType)	No
functionAliases	JSON Array (FunctionAliasType)	No
macro	JSON Array (MacroType)	No
schema	JSON Array (SchemaType)	No
storedProcedure	JSON Array (StoredProcedureType)	No

DbType

Name	JSON Data Type	Required
name	String	Yes
stagingDatabase or targetStagingDatabase Note: These parameters are interchangeable; you can use either, but not both at the same time.	String	No
sourceStagingDatabase	String	No
stagingDatabaseForTable	String	No
targetDatabase	String	No
compareDDL	Boolean	No
replaceDatabase	Boolean	No
selection	String	Yes
database	JSON Array	No
table	JSON Array (TableType)	No
map	String	No
colocate	String	No

DecoratedDSATaskType

Name	Description	JSON Data Type
type	Always DSATaskType	String
dsaJobName	The name of the DSA job	String
jobModelJson	The JSON object describing the DSA job as a string	String
dataStreams	The number of data streams allocated to this task	Integer
id	The ID of this task	Integer

DecoratedJDBCTaskType

Name	Description	JSON Data Type
type	Always JDBCTaskType	String
objectType	The type of object being moved	String
sourceParent	The source database or parent object	String
sourceTable	The table from which to read data	String
sourceOperatorType	The operator of the source database	String
sourceScripts	Scripts executed against the source database	JSON Array (String)
targetParent	The target database or parent object	String
targetTable	The destination table for data	String
targetOperatorType	The operator of the target database	String
targetScripts	Scripts executed against the target database	JSON Array (String)
dataStreams	The number of data streams allocated to this task	Integer
id	The ID of the task	Integer

DecoratedRowCountValidationTaskType

Name	Description	JSON Data Type
type	Always RowCountValidationTaskType	String
objectType	The type of object being moved	String
sourceParent	The source database or parent object	String

Name	Description	JSON Data Type
sourceTable	The table from which to read data	String
sourceOperatorType	The operator of the source database	String
sourceScripts	Scripts executed against the source database	JSON Array (String)
targetParent	The target database or parent object	String
targetTable	The destination table for data	String
targetOperatorType	The operator of the target database	String
targetScripts	Scripts executed against the target database	JSON Array (String)
dataStreams	The number of data streams allocated to this task	Integer
id	The ID of the task	Integer

DecoratedSQLTaskType

Name	Description	JSON Data Type
type	Always SQLTaskType	String
objectMovePhrase	The phrase for moving the object	String
targetParent	The target database or parent object	String
targetTable	The destination table for data	String
targetOperatorType	The operator of the target database	String
targetScripts	A list of SQL scripts executed against the target database	JSON Array (String)
dataStreams	The number of data streams allocated to this task	Integer
id	The ID of the task	Integer

DecoratedTPTAPITaskType

Name	Description	JSON Data Type
type	Always TPTAPITaskType	String
objectType	The type of object to be moved	String
sourceSessionsCount	The number of sessions allocated for the source	Integer
targetSessionsCount	The number of sessions allocated for the target	Integer

Name	Description	JSON Data Type
sourceParent	The source database or parent element	String
sourceTable	The table from which to read data	String
sourceOperatorType	The operator of the source database	String
sourceScripts	A list of scripts executed against the source database	String
targetParent	The destination database or parent element	String
targetTable	The destination table for data	String
targetOperatorType	The operator of the target database	String
targetScripts	A list of scripts executed against the target database	Integer
dataStreams	The number of data streams allocated to this task	Integer
id	The ID of this task	Integer

DSASettingsType

Name	JSON Data Type	Required
targetGroupName	String	No
parallelBuilds	Integer	No

EnforcementRequest

Name	Description	JSON Data Type	Required
enumerate	If true, lists all the permissions granted to the user for the specified resource.	Boolean	No
logic	Displays what logic to use when combining multiple permissions.	String	No
permissions	List of permissions or actions used to check if the user or role has been granted. Only one permission or action is listed.	String	Yes

EnforcementResponse

Name	Description	JSON Data Type	Required
result	True if the user has the specified permissions. False if user does not have the specified permissions.	Boolean	No
permissions	List of permissions or actions used to check if the user or role has been granted. Only one permission or action is listed.	JSON Array (PermissionResponse)	No

ForeignServerType

Name	JSON Data Type	Required
name	String	Yes
selection	String	Yes
map	String	No
colocate	String	No

FunctionAliasType

Name	JSON Data Type	Required
name	String	Yes
database	String	Yes
selection	String	Yes

IndexType

Name	JSON Data Type	Required
name	String	Yes
targetName	String	No
indexDatabase	String	Yes
targetIndexDatabase	String	No
indexType	String	Yes
copyStats	Boolean	No

Name	JSON Data Type	Required
selection	String	Yes
map	String	No
colocate	String	No

JobSecurityType

Name	JSON Data Type	Required
ownerName	String	No
readPermission	Object (PermissionType)	No
writePermission	Object (PermissionType)	No
executePermission	Object (PermissionType)	No

KeyValueStringPair

Name	Description	JSON Data Type
key	The key of the parameter set	String
value	The value of the parameter	String

LifeCycleStatus

The following table lists the REST API object type details for LifeCycleStatus:

Name	Description	JSON Data Type
startTime	Event start time	Long
endTime	Event end time	Long
eventType	Event type	String
message	Event message	String

Link

Name	Description	JSON Data Type	Required
rel	Reference link type	String	No

Name	Description	JSON Data Type	Required
href	A link to the job status, past job, or job steps	String	No

LoginType

Name	Description	JSON Data Type	Required
teradata	Teradata system	Object (TDType)	No
userPool	Pool name	String	No

MacroType

Name	JSON Data Type	Required
name	String	Yes
database	String	Yes
selection	String	Yes

Messages

Name	Description	JSON Data Type
errors	Errors that occurred processing the request	JSON Array (strings)
warnings	Warnings that occurred processing the request	JSON Array (strings)

PartialTableType

Name	JSON Data Type	Required
sqlWhereClause	String	Yes
keyColumns	JSON Array (String)	No

PermissionResponse

Name	Description	JSON Data Type	Required
resource	Name of resource.	String	No

Name	Description	JSON Data Type	Required
actions	List of actions available for the specified resource.	JSON Array (EnforcementResponse)	No

PermissionType

Name	JSON Data Type	Required
users	JSON Array (String)	No
roles	JSON Array (String)	No

Policy

Name	Description	JSON Data Type	Required
actions	A list of actions, such as read, write, and execute, that the policy grants to the principals for a specified resource. Valid actions for the resource types are: <ul style="list-style-type: none"> tdrn:datamover:daemon_access = read, write, execute tdrn:datamover:daemon_advanced = allowDSA, allowT2T, allowUpdatejobPlan, allowChangeJobPriority, allowTPTAPILOAD, allowTPTAPIUPDATE, allowTPTAPISTREAM, maxNumberOfStreams tdrn:datamover:job = read, write, execute, owner 	String	Yes
description	Policy description.	String	No
name	Policy name.	String	No
principals	List of principal names that apply to this policy. Principals must all be the same type (user, group, or role).	String	Yes
resources	List of resources that apply to this policy. Resource is in <code>tdrm:datamover:resource_name</code> format. Valid values are: <ul style="list-style-type: none"> tdrn:datamover:daemon_access:* tdrn:datamover:daemon_advanced:* tdrn:datamover:job_name 	String	Yes
service	Optional service name associated with the policy. Not required, but if provided, must be datamover .	String	No
type	The type of principal: user or role.	String	Yes

PrefixProperties

Name	Description	JSON Data Type
prefix_name	Name of the folder in the S3 bucket	String
storage_devices	Number of storage devices	Number

RestStepListType

Name	Description	JSON Data Type
dataStreams	The number of data streams allocated to this task	Integer
tasks	A list of tasks associated with this job step	JSON Array (Decorated*TaskType)
type	A descriptor of the type of step this represents	String
id	The ID of this job step	Integer

RowCountStatus

The following table lists the REST API object type details for RowCountStatus:

Name	Description	JSON Data Type
sourceDatabase	Source database name	String
sourceObject	Source object name	String
targetDatabase	Target database name	String
targetObject	Target object name	String
status	Valid row count status values: • IN_SYNC • OUT_OF_SYNC	String
validationType	Valid validation type values: • ALL • NONE • PARTIAL	String
sourceRows	Number of source object rows	Long
targetRows	Number of target object rows	Long
error	Row count error message	String

S3Properties

Name	Description	JSON Data Type
access_key_id	Access Key ID used for AWS	String
secret_access_key	Secret Access Key for AWS	String
secret_access_key_encrypted	True: secret_access_key encrypted by user False (default): secret_access_key value is plaintext	Boolean
buckets_by_region	List of S3 Buckets organized by region	JSON Array (BucketsByRegionProperties)

SchemaType

Name	JSON Data Type	Required
name	String	Yes
selection	String	Yes
compareDDL	Boolean	No

SessionsAndStreamsType

Name	Description	JSON Data Type
value	The number of sessions or streams	Integer
type	USER_DEFINED or SYSTEM_GENERATED, depending on the source of the value	String

SettingsType

Name	Description	JSON Data Type	Required
priority	Job priority	String	No
overwriteExistingObjects		Boolean	No
freezeJobSteps		Boolean	No
stagingDatabase or targetStagingDatabase		String	No

Name	Description	JSON Data Type	Required
Note: These parameters are interchangeable; you can use either, but not both at the same time.			
sourceStagingDatabase		String	No
stagingDatabaseForTable		String	No
targetDatabase		String	No
compareDDL		Boolean	No
logLevel		String	No
logToEventTable		JSON Array (String)	No
queryBand		String	No
tdTdSettings	Settings for Teradata to Teradata (bi-directional)	Object TDSettings	No
dsaSettings	Settings for DSA jobs	Object DSASettings DSASettingsType	No
dbClientEncryption		Boolean	No
map		String	No
colocate		String	No
cloudStagingArea		Object CloudStagingAreaType CloudStagingAreaType	No

SourceTargetPair

Name	Description	JSON Data Type
source_system	Source system name	String
source_system_target_group	Source system target group	String
target_system	Target system name	String
target_system_target_group	Target system target group	String

StepType

Name	Description	JSON Data Type
id	Step ID	Long
status	Status of the step	String
type	Type of step	String
startTime	Start time if step started, null if never started	String
endTime	Step end time, null if never started	String
duration	Amount of time (in seconds) taken to start step	Integer

StoredProcedureType

Name	JSON Data Type	Required
name	String	Yes
database	String	Yes
selection	String	Yes

StreamInfo

Name	Description	JSON Data Type
utility	Utility type containing the stream information	String
taskID	The task ID associated with the stream information	Long
agentName	Name of the agent where the stream information is captured	String
operator	The underlying operator of the utility	String
streamID	The stream ID for the generated stream	Long
objectName	Object name	String
bytesProcessed	Number of bytes processed	Long
processSpeed	Rate of bytes per second	Long
timestamp	Timestamp when stream information was last captured	String

SystemLevelDatabaseType

Item	Description	JSON Data Type
systemName	System name	String
stagingDatabase or targetStagingDatabase Note: These parameters are interchangeable; you can use either, but not both at the same time.	Target staging database name	String
sourceStagingDatabase	Source staging database name	String
stagingDatabaseForTable	Staging database for tables name	String
targetDatabase	Target database	String

SystemPairType

Item	Description	JSON Data Type
sourceSystem	Source system	String
targetSystem	Target system	String
allowStatsBack	Indicates if statsback is allowed	Boolean

SystemType

Item	Description	JSON Data Type
systemName	System name	String
users	Users in pool for that system	JSON Array (userType object)

TableType

Name	Description	JSON Data Type	Required
name		String	Yes
ownerName		String	No
stagingDatabase		String	No

Name	Description	JSON Data Type	Required
or targetStagingDatabase Note: These parameters are interchangeable; you can use either, but not both at the same time.			
sourceStagingDatabase		String	No
stagingDatabaseForTable		String	No
targetDatabase		String	No
targetName		String	No
validateRowCount		String	No
compareDDL		Boolean	No
ForceTargetStagingTable		Boolean	No
selection		String	Yes
exportWithoutSpool		Boolean	No
partialTableParameters	Encapsulates properties needed for partial table	Object (PartialTableType)	No
teradataParameters	Parameters relevant for Teradata to Teradata jobs only	Object (TDTableType)	No
column	Specifies columns	JSON Array (ColumnType)	No
dbClientEncryption		Boolean	No
map	Map name	String	No
colocate	Colocation name	String	No
stagingToTarget	Specifies how to move data from a staging table to a target table. Valid values are NOT_SPECIFIED, DELETE_INSERT, MERGE, INSERT_ONLY, DELETE_DISTINCT_INSERT	String	No

Name	Description	JSON Data Type	Required
useSourceStagingTable	Specifies if a job needs to use a source staging table.	Boolean	No

TaskStatus

Name	Description	JSON Data Type
id	Task ID	Long
parentName	Object parent name	String
objectName	Object name	String
movePhase	Phase of the data transfer	String
status	Task complete status	String
objectType	Type of object	String
totalRowsProcessed	Number of rows moved	Long
totalBytesProcessed	Number of bytes moved	Long
agentName	Name of the Data Mover agent that started the task	String
utility	Utility used to move the data	String
sessions	Number of sessions specified	Long
sessionsUsed	Number of sessions used	String
dataStreams	Number of streams used	Long
timestamp	Timestamp when task was started	String

TDASettings

Name	JSON Data Type	Required
dataStreams	Integer	No
sessions	Integer	No
queryTimeout	Integer	No
preserveColumnCase	Boolean	No
skipErrorRecords	Boolean	No

TDHSettings

Name	JSON Data Type	Required
forceUtility	String	No
foreignServer	String	No
fileOptions	String	No
fileOptionsDelimiter	String	No
transferMethod	String	No
batchInsertSize	Integer	No
mappers	Integer	No

TDSettings

Name	JSON Data Type	Required
forceUtility	String	No
dataStreams	Integer	No
sourceSessions	Integer	No
targetSessions	Integer	No
onlineArchive	Boolean	No
maxAgentsPerTask	Integer	No
copyStats	Boolean	No

TDTableType

Name	JSON Data Type	Required
copyStats	Boolean	No
allowTPTLoadMultiset	Boolean	No
overrideLockAccess	Boolean	No

TDType

Name	Description	JSON Data Type	Required
tdpid	Teradata system name	String	Yes
username		String	No
password		String	No
passwordEncrypted	true: password has encrypted text false: plain text (default)	Boolean	No
logonMech		String	No
logonMechData		String	No
accountId		String	No
sessionCharset		String	No

TriggerType

Name	JSON Data Type	Required
database	String	Yes
subjectTableDatabase	String	Yes
table	String	Yes
name	String	No
actionTime	Object (ActionTimeType)	Yes
selection	String	Yes

UserGroupType

Name	Description	JSON Data Type
poolName	Pool name	String
systems	Systems in the pool	JSON Array (systemType)

UserType

Name	Description	JSON Data Type
encrypted_password	Password (encrypted)	String
userName	User name	String
password	Password (text)	String

ValueType

Item	Description	JSON Data Type
value	Property value	String
system	System that applies the value (null if value applies to all systems)	String

ViewTableType

Name	JSON Data Type	Required
targetTable	String	No
targetDatabase	String	No

ViewType

Name	JSON Data Type	Required
name	String	Yes
database	String	No
viewDataTable	Object (viewTableType)	No
stagingDatabase or targetStagingDatabase Note: These parameters are interchangeable; you can use either, but not both at the same time.	String	No
sourceStagingDatabase	String	No
stagingDatabaseForTable	String	No

Name	JSON Data Type	Required
validateRowCount	String	No
compareDDL	Boolean	No
partialTable	Object (PartialTableType)	No
forceTargetStagingTable	Boolean	No
selection	String	Yes
copyData	Boolean	No
dbClientEncryption	Boolean	No

Data Mover XML Schemas

Data Mover XML Schemas

```
<?xml version="1.0" encoding="UTF-8"?>

<!--
Copyright (C) 2009-2015 by Teradata Corporation.
All Rights Reserved.
TERADATA CORPORATION CONFIDENTIAL AND TRADE SECRET
-->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://schemas.teradata.com/dataMover/v2009"
xmlns="http://schemas.teradata.com/dataMover/v2009"
            elementFormDefault="qualified" attributeFormDefault="unqualified">

    <!-- dmMessageBase - The base for all DM messages -->
    <xsd:complexType name="dmMessageBase">
        <xsd:sequence>
            <!--
                Source identifier. Source should use set message selector
to collect replies
                with correlationId set to this value
                -->
                <xsd:element name="source" type="xsd:string"
default="unidentified" minOccurs="0" />
            </xsd:sequence>
            <!--
                The schema version is required and should match the schema
file used
                to define the xml
                -->
                <xsd:attribute name="dmVersion" type="xsd:string"
                    use="optional" />
        </xsd:complexType>

    <!-- dmCommandBase - The base for all commands into DM -->
    <xsd:complexType name="dmCommandBase">
        <xsd:complexContent>
            <xsd:extension base="dmMessageBase">
```



```

        <xsd:sequence>
            <!--
                Optional user-defined correlation
                id that will be sent back with the response
            -->
            <xsd:element name="id" type="xsd:long"
                default="0" minOccurs="0" />
            <xsd:element name="dmSecurity"
                type="dmCheckSecurityType" minOccurs="0" />

            <!-- System - host name - that is
                initiating the commandLine request -->
            <xsd:element name="host_name"
                type="xsd:string"
                minOccurs="0" maxOccurs="1"
                default="undefined" />

            <!-- System - IP address - that is
                initiating the commandLine request -->
            <xsd:element name="host_address"
                type="xsd:string"
                minOccurs="0" maxOccurs="1"
                default="undefined" />
        </xsd:sequence>
    </xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<!-- Define Command Header -->
<xsd:complexType name="commandHeaderType">
    <xsd:sequence>
        <xsd:element name="command_action" type="xsd:string" />
        <xsd:element name="message" minOccurs="0"
            type="xsd:string" />
    </xsd:sequence>
</xsd:complexType>

<!-- Define Output Header -->
<xsd:complexType name="warningType">
    <xsd:complexContent>
        <xsd:extension base="problemType">
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

```

```

<xsd:complexType name="errorType">
  <xsd:complexContent>
    <xsd:extension base="problemType">
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="problemType">
    <xsd:sequence>
      <xsd:element name="code" type="xsd:string" />
      <xsd:element name="message" type="xsd:string" />
      <xsd:element name="data" type="xsd:string"
        nillable="true" minOccurs="0"
maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="outputHeaderType">
    <xsd:complexContent>
      <xsd:extension base="commandHeaderType">
        <xsd:sequence>
          <xsd:element name="id" type="xsd:long"
default="0" />
          <xsd:element name="status"
type="xsd:boolean" default="true" />
          <!-- errors -->
          <xsd:element name="error"
type="errorType" minOccurs="0"
maxOccurs="unbounded" />
          <!-- warnings -->
          <xsd:element name="warning"
type="warningType"
minOccurs="0"
maxOccurs="unbounded" />
          <!-- daemonID -->
          <xsd:element name="daemonID" type="xsd:string"
minOccurs="0" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <!-- dmOutputBase - The base for all Daemon response messages to commands

```

```

-->
    <xsd:complexType name="dmOutputBase">
        <xsd:complexContent>
            <xsd:extension base="dmMessageBase">
                <xsd:sequence>
                    <!-- Command Header -->
                    <xsd:element name="commandHeader"
minOccurs="0" type="outputHeaderType" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <!-- dmAgentCommandBase - The base for all Agent to Daemon messages -->
    <xsd:complexType name="dmAgentCommandBase">
        <xsd:complexContent>
            <xsd:extension base="dmMessageBase">
                <xsd:sequence>
                    <!-- Command Header -->
                    <xsd:element name="commandHeader"
minOccurs="0" type="commandHeaderType" />
                    <xsd:element name="id" type="xsd:long"
default="0" minOccurs="0" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <!-- dmAck - for acknowledgment information -->
    <xsd:element name="dmAck">
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="dmOutputBase">
                    <xsd:sequence>
                        <xsd:element
name="securityEnabled" type="xsd:boolean"
default="false"
maxOccurs="1" minOccurs="0" />
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>

```

```

    <!-- DMQuery -->
    <xsd:element name="dmQuery">
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="dmCommandBase">
                    <xsd:sequence>

Teradata database -->
nillable="false" type="xsd:string" />

Checks, Source Teradata log on id -->
minOccurs="0" type="xsd:string" />

123456789, (optional)Source Teradata log on
name="source_password" minOccurs="0"

y1rX3DXZNz (optional)Encrypted source
name="source_password_encrypted"
type="xsd:string" />

NTLM, (optional)Logon mechanism to use
system
name="source_logon_mechanism" minOccurs="0"

    <!-- source_tdpid Checks, Source
    <xsd:element name="source_tdpid"

    <!-- source_user TD_API_user,
    <xsd:element name="source_user"

    <!--
        source_password
        password
    -->
    <xsd:element
        type="xsd:string" />
    <!--
        source_password_encrypted
        Teradata log on password
    -->
    <xsd:element
        minOccurs="0"

    <!--
        source_logon_mechanism
        when logging on to source
    -->
    <xsd:element
        type="xsd:string" />
    <!--

```

```

source_logon_mechanism_data joe@domain1 @@mypassword
                                (optional)Additional
parameters needed for the logon mechanism
                                being used
                                -->
                                <xsd:element
name="source_logon_mechanism_data"
                                minOccurs="0"
type="xsd:string" />
                                <xsd:element
name="source_account_id" minOccurs="0"
                                type="xsd:string" />
                                <!-- querying only a specific
database -->
                                <xsd:element name="database"
minOccurs="0" type="xsd:string" />
                                <!--
                                dir /user/tptapi/lists,
                                list.
                                -->
                                <xsd:element name="dir"
minOccurs="0" type="xsd:string" />
                                <!--
                                object_list
                                object list
                                -->
                                <xsd:element name="object_list"
minOccurs="0" type="xsd:string" />
                                <!--
                                database db1
                                databases can be
                                different databases
                                -->
                                <xsd:element
name="limit_to_database" minOccurs="0"
                                maxOccurs="unbounded"
type="xsd:string" />
                                <!--

```

```

    (optional)Limit query to objects in this
    Multiple replication groups can be specified
    option with different replication groups

    name="limit_to_replication_group"
    maxOccurs="unbounded" type="xsd:string" />
    name="response_timeout" minOccurs="0"
    default="10" />
    name="setQueryBand" minOccurs="0"
    default="false" />
    name="queryBand" minOccurs="0"

    </xsd:sequence>
  </xsd:extension>
</xsd:complexContent>
</xsd:complexType>
</xsd:element>

<!-- dmDatabaseVersion -->
<xsd:element name="dmDatabaseVersion">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="dmCommandBase">
        <xsd:sequence>

Teradata database -->
nillable="false" type="xsd:string" />

Checks, Source Teradata log on id -->
nillable="false" type="xsd:string" />

    replication_group gp1,
    replication group.
    by repeating the same
  -->
  <xsd:element
    minOccurs="0"
    <xsd:element
      type="xsd:int"
    <xsd:element
      type="xsd:boolean"
    <xsd:element
      type="xsd:string" />
  </xsd:sequence>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>
</xsd:element>

<!-- dmDatabaseVersion -->
<xsd:element name="dmDatabaseVersion">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="dmCommandBase">
        <xsd:sequence>

    <!-- source_tdpid Checks, Source
    <xsd:element name="source_tdpid"

    <!-- source_user TD_API_user,
    <xsd:element name="source_user"

```

123456789, (optional)Source Teradata log on	<!-- source_password
	password
	-->
name="source_password" minOccurs="0"	<xsd:element
	type="xsd:string" />
	<!--
y1rX3DXZNz (optional)Encrypted source	source_password_encrypted
	Teradata log on password
	-->
name="source_password_encrypted"	<xsd:element
	minOccurs="0"
type="xsd:base64Binary" />	
	<!--
NTLM, (optional)Logon mechanism to use	source_logon_mechanism
	when logging on to source
system	
	-->
name="source_logon_mechanism" minOccurs="0"	<xsd:element
	type="xsd:string" />
	<!--
source_logon_mechanism_data joe@domain1 @@mypassword	(optional)Additional
parameters needed for the logon mechanism	being used
	-->
name="source_logon_mechanism_data"	<xsd:element
	minOccurs="0"
type="xsd:string" />	
name="source_account_id" minOccurs="0"	<xsd:element
	type="xsd:string" />
name="response_timeout" minOccurs="0"	<xsd:element
default="10" />	type="xsd:int"

```

        </xsd:sequence>
    </xsd:extension>
</xsd:complexContent>
</xsd:complexType>
</xsd:element>

<!-- Define output from dmDatabaseVersion -->
<xsd:element name="dmDatabaseVersionOutput">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmOutputBase">
                <xsd:sequence>
                    <xsd:element name="version"
nillable="false" type="xsd:string" />
                    <xsd:element name="majorVersion"
nillable="false"
                                type="xsd:int" />
                    <xsd:element name="minorVersion"
nillable="false"
                                type="xsd:int" />
                    <xsd:element
name="maintenanceVersion" nillable="false"
                                type="xsd:int" />
                    <xsd:element name="efixVersion"
nillable="false" type="xsd:int" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<xsd:element name="databaseOutput">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmOutputBase">
                <xsd:sequence>
                    <!-- Databases to be copied -->
                    <xsd:element name="database"
minOccurs="0"
                                maxOccurs="unbounded"
                                type="databaseType" />
                    <!-- Triggers to be copied -->
                    <xsd:element ref="triggers"
minOccurs="0" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

```



```

        <!-- Indices to be copied -->
        <xsd:element ref="indices"
minOccurs="0" />

        <!-- Views to be copied -->
        <xsd:element ref="views"
minOccurs="0" />

        <!-- Journals to be copied -->
        <xsd:element ref="journals"
minOccurs="0" />

        <!-- Macros to be copied -->
        <xsd:element ref="macros"
minOccurs="0" />

        <!-- SQL SP to be copied -->
        <xsd:element
ref="stored_procedures" minOccurs="0" />
    </xsd:sequence>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>
</xsd:element>

<!-- Valid Selection values for objects -->
<xsd:simpleType name="selection">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="included" />
        <xsd:enumeration value="excluded" />
        <!-- selection="all" is valid only for Databases -->
        <xsd:enumeration value="all" />
        <xsd:enumeration value="unselected" />
    </xsd:restriction>
</xsd:simpleType>

<!-- Cloud Staging Area Type -->
<xsd:complexType name="cloudStagingArea">
    <xsd:sequence>
        <xsd:element name="name" minOccurs="0"
type="xsd:string" />
    </xsd:sequence>
</xsd:complexType>

<!-- Database -->
<xsd:complexType name="databaseType">
    <xsd:sequence>

```

```

        <xsd:element name="name" minOccurs="1" maxOccurs="1"
            type="xsd:string" />
        <!-- option to use staging database -->

        <xsd:element name="source_staging_database" type="targetDatabaseType"
maxOccurs="1" minOccurs="0" />
        <!-- target_staging_database new tag used to replace
staging_database -->
        <xsd:element name="target_staging_database"
type="targetDatabaseType" minOccurs="0" maxOccurs="1" />
        <!-- target_staging_database_for_table : new tag used to
replace staging_database_for_table -->

        <xsd:element name="staging_database"
type="targetDatabaseType" minOccurs="0" maxOccurs="1" />
        <xsd:element name="staging_database_for_table"
type="targetDatabaseType" minOccurs="0" maxOccurs="1" /
>
        <xsd:element name="target_database"
type="targetDatabaseType" maxOccurs="1" minOccurs="0" />
        <!-- compare DDL of the source tables in the database
with the target database tables -->
        <xsd:element name="compare_ddl" minOccurs="0"
maxOccurs="1"
            type="triStateType" default="unspecified" />
        <xsd:element name="map" minOccurs="0" maxOccurs="1"
type="xsd:string" />
        <xsd:element name="colocate" minOccurs="0" maxOccurs="1"
type="xsd:string" />
        <!-- compare journaling forward if table uses journaling
-->
        <xsd:element name="journaling" minOccurs="0"
maxOccurs="1"
            type="triStateType" default="unspecified" />
        <xsd:element name="database" minOccurs="0"
maxOccurs="unbounded" type="databaseType" />
        <xsd:element name="table" minOccurs="0"
maxOccurs="unbounded" type="tableType" />

    </xsd:sequence>
    <xsd:attribute name="replaceDatabase" type="triStateType"
use="optional"/>
    <xsd:attribute name="selection" type="selection"
use="required" />
</xsd:complexType>

```

```

<!-- Valid force utility values -->
<xsd:simpleType name="force_utility">
  <xsd:restriction base="xsd:string">
    <xsd:pattern
      value="[Aa][Rr][Cc]|[Dd][Ss][Aa]|[Tt][Pp][Tt][Aa]
[Pp][Ii]|[Tt][Pp][Tt][Aa][Pp][Ii]_[Ll][Oo][Aa][Dd]|[Tt][Pp][Tt][Aa][Pp][Ii]_[Uu]
[Pp][Dd][Aa][Tt][Ee]" />
    <xsd:pattern
      value="[Tt][Pp][Tt][Aa][Pp][Ii]_[Ss][Tt][Rr][Ee]
[Aa][Mm]|[Jj][Dd][Bb][Cc]|[Ss][Qq][Ll][Hh]|[Hh][Aa][Dd][Oo][Oo][Pp]_[Cc][Oo][Nn]
[Nn][Ee][Cc][Tt][Oo][Rr]|[Tt][2][Tt]" />
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="booleanType">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="[Tt][Rr][Uu][Ee]|[Ff][Aa][Ll][Ss]
[Ee]"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="triStateType">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="[Tt][Rr][Uu][Ee]|[Ff][Aa][Ll][Ss][Ee]|
[Uu][Nn][Ss][Pp][Ee][Cc][Ii][Ff][Ii][Ee][Dd]"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="RowCountValidationType">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="[Nn][Oo][Nn][Ee]|[Pp][Aa][Rr][Tt][Ii]
[Aa][Ll]|[Aa][Ll][Ll]"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="StagingToTargetType">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="[Nn][Oo][Tt]_[Ss][Pp][Ee][Cc][Ii][Ff]
[Ii][Ee][Dd]|[Dd][Ee][Ll][Ee][Tt][Ee]_[Ii][Nn][Ss][Ee][Rr][Tt]|[Mm][Ee][Rr][Gg]
[Ee]|[Ii][Nn][Ss][Ee][Rr][Tt]_[Oo][Nn][Ll][Yy]|[Dd][Ee][Ll][Ee][Tt][Ee]_[Dd][Ii]
[Ss][Tt][Ii][Nn][Cc][Tt]_[Ii][Nn][Ss][Ee][Rr][Tt]"/>
  </xsd:restriction>
</xsd:simpleType>

```

```

    <xsd:simpleType name="createModeType">
      <xsd:restriction base="xsd:string">
        <xsd:pattern value="[Cc][Rr][Ee][Aa][Tt][Ee]|[Mm][Oo][Vv]
[Ee]"/>
      </xsd:restriction>
    </xsd:simpleType>

    <!-- Whether a job is static (job scripts has been modified) or dynamic
(create/execute again) -->
    <xsd:simpleType name="jobType">
      <xsd:restriction base="xsd:string">
        <xsd:pattern value="STATIC|DYNAMIC" />
      </xsd:restriction>
    </xsd:simpleType>

    <!-- Job Priority Level Type -->
    <xsd:simpleType name="jobPriorityType">
      <xsd:restriction base="xsd:string">
        <xsd:pattern value="[Ll][Oo][Ww]|[Mm][Ee][Dd][Ii][Uu][Mm]|
[Hh][Ii][Gg][Hh]|[Uu][Nn][Ss][Pp][Ee][Cc][Ii][Ff][Ii][Ee][Dd]" />
      </xsd:restriction>
    </xsd:simpleType>

    <!-- Job Definition -->
    <xsd:complexType name="jobDefinitionType">
      <xsd:complexContent>
        <xsd:extension base="dmCommandBase">
          <xsd:sequence>
            <!--
              job_name JOB1DAILY, (optional)
              job name for this job, must be
              specified
              unique. Auto-generated if not
              -->
            <xsd:element name="job_name"
minOccurs="0" type="xsd:string" />

            <!-- internally use to indicate if this is
create or move job -->
            <xsd:element name="create_mode"
minOccurs="0" type="createModeType" />

            <!-- internally use to indicate job

```

```

priority level -->
minOccurs="0" type="jobPriorityType" />

<xsd:element name="job_priority"

user modified the job (static)
during runtime (dynamic) -->
minOccurs="0" type="jobType" />

<!-- internally use to determine whether
or should be recreated

<xsd:element name="job_type"

database -->
minOccurs="0" type="xsd:string" />

<!-- source_tdpid Checks, Source Teradata

<xsd:element name="source_tdpid"

Source Teradata log on id -->
minOccurs="0" type="xsd:string" />

<!-- source_user TD_API_user, Checks,

<xsd:element name="source_user"

<!--
source_password 123456789,
password
-->
<xsd:element name="source_password"

type="xsd:string" />
<!--
source_password_encrypted
Teradata log on password
-->
<xsd:element
minOccurs="0"

<xsd:element name="source_userid_pool"

type="xsd:string" />
<!--
source_logon_mechanism NTLM,
(optional)Logon mechanism to use when

```

```

                                logging on to source system
                                -->
                                <xsd:element
name="source_logon_mechanism" minOccurs="0"
                                type="xsd:string" />
                                <!--
                                source_logon_mechanism_data
                                (optional)Additional parameters
                                needed for the logon mechanism
                                being used
                                -->
                                <xsd:element
name="source_logon_mechanism_data"
                                minOccurs="0"
                                type="xsd:string" />
                                <xsd:element name="source_account_id"
                                type="xsd:string" />
                                <!-- source_session_charset Source
Teradata session character set -->
                                <xsd:element
name="source_session_charset" minOccurs="0" type="xsd:string" />
                                <!-- source Aster system if any -->
                                <xsd:element name="source_aster_system"
minOccurs="0" maxOccurs="1" type="asterSystemType" />
                                <!-- target_dbs Checks, Source Teradata
database -->
                                <xsd:element name="target_tdpid"
minOccurs="0" type="xsd:string" />
                                <!-- target_user TD_API_user, Source
Teradata log on id -->
                                <xsd:element name="target_user"
minOccurs="0" type="xsd:string" />
                                <!--
                                source_password 123456789,
                                password
                                -->
                                <xsd:element name="target_password"

```

```

minOccurs="0"

                                type="xsd:string" />
                                <!--
                                source_password_encrypted
                                Teradata log on password
                                -->
                                <xsd:element
                                minOccurs="0"
                                <xsd:element name="target_userid_pool"
                                type="xsd:string" />
                                <!--
                                source_logon_mechanism NTLM,
                                logging on to source system
                                -->
                                <xsd:element
                                name="target_logon_mechanism" minOccurs="0"
                                type="xsd:string" />
                                <!--
                                source_logon_mechanism_data
                                (optional)Additional parameters
                                being used
                                -->
                                <xsd:element
                                minOccurs="0"
                                <xsd:element name="target_account_id"
                                type="xsd:string" />
                                <!-- target_session_charset Source
                                Teradata session character set -->
                                <xsd:element
                                name="target_session_charset" minOccurs="0" type="xsd:string" />
                                <!--
                                use_userpool true,

```

```

(optional)weather to use the
                                pool of target user Ids
-->
<xsd:element name="use_userid_pool"
                                type="booleanType" />
minOccurs="0"
                                <xsd:element name="group_userid_pool"
                                type="xsd:string" />
minOccurs="0"
                                <!-- target Aster system if any -->
                                <xsd:element name="target_aster_system"
minOccurs="0" maxOccurs="1" type="asterSystemType" />
                                <xsd:element name="data_streams"
                                <xsd:element name="source_sessions"
                                <xsd:element name="target_sessions"
                                <xsd:element name="max_agents_per_task"
                                <xsd:element name="response_timeout"
                                type="xsd:int" default="10" />
                                <!--
                                sync this option is only used by
                                when set to true, command line
                                before exiting
-->
                                <xsd:element name="sync"
                                maxOccurs="1" minOccurs="0" />
                                <xsd:element
                                minOccurs="0" type="triStateType"
                                <xsd:element name="freeze_job_steps"
                                minOccurs="0" type="triStateType"
                                <xsd:element name="overwrite_existing_objects"
                                default="unspecified"/>
                                <xsd:element name="freeze_job_steps"
                                minOccurs="0" type="triStateType"
                                default="unspecified" />

```



```

                                <!--
                                force_utility (optional) force
Data Mover to use the specified
                                utility (dsa, t2t, arc, tptapi,
tptapi_load, tptapi_update, tptapi_stream, jdbc)
                                -->
                                <xsd:element name="force_utility"
minOccurs="0"
                                type="force_utility" />

                                <xsd:element name="source_staging_database"
type="targetDatabaseType" maxOccurs="1" minOccurs="0" />
                                <!-- target_staging_database new tag used
to replace staging_database -->
                                <xsd:element
name="target_staging_database" type="targetDatabaseType"  minOccurs="0"
maxOccurs="1" />
                                <!-- target_staging_database_for_table :
new tag used to replace staging_database_for_table -->
                                <xsd:element name="staging_database"
type="targetDatabaseType"
                                minOccurs="0" maxOccurs="1" />
                                <xsd:element
name="staging_database_for_table" type="targetDatabaseType"
                                minOccurs="0" maxOccurs="1" />
                                <xsd:element name="target_database"
type="targetDatabaseType"
                                maxOccurs="1" minOccurs="0" />
                                <xsd:element name="use_foreign_server"
type="useForeignServerType"
                                maxOccurs="1" minOccurs="0" />

                                <xsd:element name="compare_ddl"
minOccurs="0" maxOccurs="1"
                                type="triStateType"
                                default="unspecified" />

                                <xsd:element name="log_level"
minOccurs="0" type="xsd:int"
                                default="0" />

                                <!-- online archive -->

```

```

minOccurs="0"
                                <xsd:element name="online_archive"
                                type="triStateType"
                                default="unspecified"/>

                                <xsd:element name="log_to_event_table"
                                minOccurs="0"
                                maxOccurs="unbounded"
                                type="xsd:string" />
                                <xsd:element name="map" minOccurs="0"
                                maxOccurs="1" type="xsd:string" />
                                <xsd:element name="colocate" minOccurs="0"
                                maxOccurs="1" type="xsd:string" />
                                <xsd:element
                                name="enable_incremental_restore" minOccurs="0" type="triStateType"
                                default="unspecified" />
                                <xsd:element name="dsa_options"
                                minOccurs="0" type="dsaOptionsType" />
                                <!-- Aster Job level parameters. -->
                                <xsd:element name="aster_options"
                                minOccurs="0" type="asterOptionType" />
                                <!-- Cloud Staging Area -->
                                <xsd:element name="cloud_staging_area"
                                minOccurs="0" type="cloudStagingArea" />

                                <!-- Databases to be copied -->
                                <xsd:element name="db_client_encryption"
                                minOccurs="0" type="triStateType" default="unspecified"/>
                                <xsd:element name="copy_stats"
                                minOccurs="0" maxOccurs="1" type="triStateType" default="unspecified"/>
                                <xsd:element name="database"
                                minOccurs="0"
                                maxOccurs="unbounded"
                                type="databaseType" />

                                <!-- Triggers to be copied -->
                                <xsd:element ref="triggers"
                                minOccurs="0" />

                                <!-- Indices to be copied -->
                                <xsd:element ref="indices"
                                minOccurs="0" />

                                <!-- Views to be copied -->

```

```

                                <xsd:element ref="views" minOccurs="0" />

                                <!-- Views to be copied -->
                                <xsd:element ref="foreign_servers"
minOccurs="0" />

                                <!-- Function Aliases to be moved -->
                                <xsd:element ref="function_aliases"
minOccurs="0" />

                                <!-- Journals to be copied -->
                                <xsd:element ref="journals"
minOccurs="0" />

                                <!-- Macros to be copied -->
                                <xsd:element ref="macros"
minOccurs="0" />

                                <!-- Schemas to be copied -->
                                <xsd:element ref="schemas"
minOccurs="0" />

                                <!-- Procedures to be copied -->
                                <xsd:element ref="stored_procedures"
minOccurs="0" />

                                <!-- Functions to be copied -->
                                <xsd:element ref="functions" minOccurs="0" />

                                <!-- optional unit of work id -->
                                <xsd:element name="uowid" minOccurs="0"
type="xsd:string" />

                                <!-- optional query band parameter -->
                                <xsd:element name="query_band"
minOccurs="0" type="xsd:string" />

                                <!-- optional enable CLI or TPT trace logging -->
                                <xsd:element name="enable_trace_log" type="traceLogType"
minOccurs="0" maxOccurs="1"/>

                                <!-- optional arc parameter -->
                                <xsd:element
name="additional_arc_parameters" minOccurs="0" type="xsd:string" />

```

```

        </xsd:sequence>
    </xsd:extension>
</xsd:complexContent>
</xsd:complexType>

    <!-- Changed Job Definition, all the properties defined in
JobDefinitionType are optional here -->
    <xsd:complexType name="changedJobDefinitionType">
        <xsd:complexContent>
            <xsd:extension base="dmCommandBase">
                <xsd:sequence>
                    <!--
                                job_name JOB1DAILY,
(optional) job name for this job, must be
                                unique.
-->
                    <xsd:element name="job_name"
minOccurs="0" type="xsd:string" />

                    <!-- internally use to indicate
                                <xsd:element name="create_mode"
if this is create or move job -->
minOccurs="0" type="createModeType" />

                    <!-- internally use to indicate
                                <xsd:element name="job_priority"
job priority level -->
minOccurs="0" type="jobPriorityType" />

                    <!-- internally use to determine
                                or should be
                                <xsd:element name="job_type"
whether user modified the job (static)
recreated during runtime (dynamic) -->
minOccurs="0" type="jobType" />

                    <!-- source_tdpid Checks, Source
                                <xsd:element name="source_tdpid"
Teradata database -->
minOccurs="0" type="xsd:string" />

                    <!-- source_user TD_API_user,

```

Checks, Source Teradata log on id -->

minOccurs="0" type="xsd:string" />

123456789, (optional)Source Teradata log on

name="source_password" minOccurs="0"

y1rX3DXZNz (optional)Encrypted source

name="source_password_encrypted"

type="xsd:string" />

name="source_userid_pool" minOccurs="0"

NTLM, (optional)Logon mechanism to use when
system

name="source_logon_mechanism" minOccurs="0"

source_logon_mechanism_data joe@domain1 @@mypassword

parameters needed for the logon mechanism

name="source_logon_mechanism_data"

type="xsd:string" />

<xsd:element name="source_user"

<!--

source_password

password

-->

<xsd:element

type="xsd:string" />

<!--

source_password_encrypted

Teradata log on password

-->

<xsd:element

minOccurs="0"

<xsd:element

type="xsd:string" />

<!--

source_logon_mechanism

logging on to source

-->

<xsd:element

type="xsd:string" />

<!--

(optional)Additional

being used

-->

<xsd:element

minOccurs="0"

```

name="source_account_id" minOccurs="0"

Source Teradata session character set -->
name="source_session_charset" minOccurs="0"

>
name="source_aster_system" minOccurs="0"
type="asterSystemType" />

Teradata database -->
minOccurs="0" type="xsd:string" />

Source Teradata log on id -->
minOccurs="0" type="xsd:string" />

123456789, (optional)Source Teradata log on

name="target_password" minOccurs="0"

y1rX3DXZNz (optional)Encrypted source

name="target_password_encrypted"
type="xsd:string" />

<xsd:element
    type="xsd:string" />

<!-- source_session_charset
<xsd:element
    type="xsd:string" />

<!-- source Aster system if any --
<xsd:element
    minOccurs="1"

<!-- target_dbs Checks, Source
<xsd:element name="target_tdpid"

<!-- target_user TD_API_user,
<xsd:element name="target_user"

<!--
    source_password
    password
-->
<xsd:element
    type="xsd:string" />

<!--
    source_password_encrypted
    Teradata log on password
-->
<xsd:element
    minOccurs="0"

<xsd:element

```

```

name="target_userid_pool" minOccurs="0"
                                type="xsd:string" />
                                <!--
                                source_logon_mechanism
                                logging on to source
                                -->
                                <xsd:element
name="target_logon_mechanism" minOccurs="0"
                                type="xsd:string" />
                                <!--
                                source_logon_mechanism_data joe@domain1 @@mypassword
                                (optional)Additional
                                parameters needed for the logon mechanism
                                being used
                                -->
                                <xsd:element
name="target_logon_mechanism_data"
                                minOccurs="0"
                                type="xsd:string" />
                                <xsd:element
name="target_account_id" minOccurs="0"
                                type="xsd:string" />
                                <!-- target_session_charset
                                Source Teradata session character set -->
                                <xsd:element
name="target_session_charset" minOccurs="0" type="xsd:string" />
                                <!--
                                use_userpool true,
                                pool of target user Ids
                                -->
                                <xsd:element
name="use_userid_pool" minOccurs="0"
                                type="booleanType" />
                                <xsd:element name="group_userid_pool"
                                type="xsd:string" />
                                <!-- target Aster system if any

```

```

-->
name="target_aster_system" minOccurs="0"
type="asterSystemType" />
                                <xsd:element
                                maxOccurs="1"

                                <xsd:element name="data_streams"
minOccurs="0" type="xsd:string"/>
                                <xsd:element
name="source_sessions" minOccurs="0" type="xsd:string"/>
                                <xsd:element
name="target_sessions" minOccurs="0" type="xsd:string"/>
                                <xsd:element
name="max_agents_per_task" minOccurs="0" type="xsd:string"/>
                                <xsd:element
name="response_timeout" minOccurs="0"
                                type="xsd:int"
                                default="10" />
                                <!--
                                sync this option is only
                                used by the command line interface
                                when set to true, command
                                line will wait for the job to complete
                                before exiting
-->
                                <xsd:element name="sync"
                                maxOccurs="1"

                                <xsd:element
                                minOccurs="0"

                                <xsd:element
                                minOccurs="0"

                                <!--
                                force_utility (optional)
                                force Data Mover to use the specified
                                utility (dsa, tpt, arc,

```



```

tptapi, tptapi_load, tptapi_update, tptapi_stream, jdbc)
-->
<xsd:element name="force_utility"
minOccurs="0"
type="force_utility" />

<xsd:element name="source_staging_database"
type="targetDatabaseType" maxOccurs="1" minOccurs="0" />
<!-- target_staging_database new
tag used to replace staging_database -->
<xsd:element
name="target_staging_database" type="targetDatabaseType" minOccurs="0"
maxOccurs="1" />

<xsd:element
name="staging_database" type="targetDatabaseType"
minOccurs="0"
maxOccurs="1" />

<xsd:element
name="staging_database_for_table" type="targetDatabaseType"
minOccurs="0"
maxOccurs="1" />

<xsd:element name="target_database"
type="targetDatabaseType"
maxOccurs="1" minOccurs="0" />
<xsd:element
name="use_foreign_server" type="useForeignServerType"
maxOccurs="1" minOccurs="0" />

<xsd:element name="compare_ddl"
minOccurs="0" maxOccurs="1"
type="triStateType"
default="unspecified"/>

<xsd:element name="log_level"
minOccurs="0" type="xsd:int"/>

<!-- online archive -->
<xsd:element
name="online_archive" minOccurs="0"
type="triStateType"
default="unspecified" />

<xsd:element

```

```

name="log_to_event_table" minOccurs="0"
maxOccurs="unbounded"
type="xsd:string" />

<xsd:element name="map" minOccurs="0" maxOccurs="1" type="xsd:string" />
<xsd:element name="colocate" minOccurs="0"
maxOccurs="1" type="xsd:string" />
<xsd:element
name="enable_incremental_restore" minOccurs="0" type="triStateType"
default="unspecified"/>
<xsd:element name="dsa_options"
minOccurs="0"
type="dsaOptionsType" />

-->
<!-- Aster Job level parameters.
<xsd:element name="aster_options"
type="asterOptionType" />

<!-- Cloud Staging Area -->
<xsd:element
name="cloud_staging_area" minOccurs="0" type="cloudStagingArea" />

<xsd:element
name="db_client_encryption" minOccurs="0" type="triStateType"
default="unspecified"/>
<xsd:element name="copy_stats"
minOccurs="0" maxOccurs="1" type="triStateType" default="unspecified"/>
<!-- Databases to be changed -->
<xsd:element name="database"
minOccurs="0"
maxOccurs="unbounded"
type="databaseType" />

<!-- Triggers to be changed -->
<xsd:element ref="triggers"
minOccurs="0" />

<!-- Indices to be changed -->
<xsd:element ref="indices"
minOccurs="0" />

<!-- Views to be changed -->

```

```

minOccurs="0" />

changed -->

ref="foreign_servers" minOccurs="0" />

-->

ref="function_aliases" minOccurs="0" />

minOccurs="0" />

<!-- Macros to be changed -->
minOccurs="0" />

<!-- Schemas to be changed -->
minOccurs="0" />

<!-- Procedures to be changed --
>

ref="stored_procedures" minOccurs="0" />

<!-- Functions to be changed -->
<xsd:element ref="functions" minOccurs="0" />

<!-- optional unit of work id -->
minOccurs="0" type="xsd:string" />

parameter -->

minOccurs="0" type="xsd:string" />

<!-- optional enable CLI or TPT trace logging -->
<xsd:element name="enable_trace_log" type="traceLogType"
<xsd:element ref="views"

<!-- Foreign Servers to be

<xsd:element

<!-- Function Aliases to be moved

<xsd:element

<!-- Journals to be changed -->
<xsd:element ref="journals"

<xsd:element ref="macros"

<xsd:element ref="schemas"

<xsd:element

```

```

minOccurs="0" maxOccurs="1"/>

                                <xsd:element
name="additional_arc_parameters" minOccurs="0" type="xsd:string" />
                                <!-- for internal use only      --
>
                                <xsd:element name="restart"
minOccurs="0"
                                type="xsd:boolean"
default="false" />
                                <xsd:element name="move"
minOccurs="0"
                                maxOccurs="1"
                                type="xsd:boolean" />
                                </xsd:element>
                                <xsd:element
name="variableChanged" type="jobVariablesChangeType"
                                minOccurs="0"
                                maxOccurs="1" />
                                <xsd:element name="saveChanges"
minOccurs="0"
                                maxOccurs="1" />
                                </xsd:sequence>
                                </xsd:extension>
                                </xsd:complexContent>
                                </xsd:complexType>
                                <!-- dmBackup -->
                                <xsd:element name="dmBackupDaemon">
                                    <xsd:complexType>
                                        <xsd:complexContent>
                                            <xsd:extension base="dmCommandBase">
                                                <xsd:sequence>
                                                    <xsd:element
name="backup_target_dir" minOccurs="0" type="xsd:string" />
                                                    </xsd:sequence>
                                                </xsd:extension>
                                            </xsd:complexContent>
                                        </xsd:complexType>
                                    </xsd:element>

                                <!-- dmBackupOutput -->
                                <xsd:element name="dmBackupDaemonOutput">

```

```

        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="dmOutputBase">
                    <xsd:sequence>
                        <xsd:element
name="backup_daemon_status" nillable="false" type="xsd:string" />
                        <xsd:element
name="backup_daemon_result" nillable="false" type="xsd:boolean" />
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>

    <!-- dmRestore -->
    <xsd:element name="dmRestoreDaemon">
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="dmCommandBase">
                    <xsd:sequence>
                        <xsd:element
name="backup_target_dir" nillable="false" type="xsd:string" />
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>

    <!-- dmRestoreOutput -->
    <xsd:element name="dmRestoreDaemonOutput">
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="dmOutputBase">
                    <xsd:sequence>
                        <xsd:element
name="restore_daemon_status" nillable="false" type="xsd:string" />
                        <xsd:element
name="restore_daemon_result" nillable="false" type="xsd:boolean" />
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>

```

```

<!-- dmCreate -->
<xsd:element name="dmCreate">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="jobDefinitionType">
        <xsd:sequence>
          <!-- permission setting for the
created job -->
          <xsd:element name="job_security"
minOccurs="0" type="securityType" />
          <xsd:element
name="command_start_time" minOccurs="0" type="xsd:dateTime" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<!-- dmCreateOutput -->
<xsd:element name="dmCreateOutput">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="dmOutputBase">
        <xsd:sequence>
          <!-- job_name, 12315DFHJKS, Job ID of the job to
be started -->
          <xsd:element name="job_name"
nillable="false" type="xsd:string" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<!-- dmStop -->
<xsd:element name="dmStop">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="dmCommandBase">
        <xsd:sequence>
          <!-- job_name, 12315DFHJKS, Job ID of the job to
be started -->
          <xsd:element name="job_name"
nillable="false" type="xsd:string" />

```

```

                                <xsd:element
name="response_timeout" minOccurs="0"
                                type="xsd:int"
default="10" />
                                </xsd:sequence>
                                </xsd:extension>
                                </xsd:complexContent>
                                </xsd:complexType>
                                </xsd:element>

<!-- dmStopOutput -->
<xsd:element name="dmStopOutput">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmOutputBase">
                <xsd:sequence>
                    <xsd:element name="job_name"
nillable="false" type="xsd:string" />
                    <xsd:element
name="job_definition_id" type="xsd:long" />
                    <!-- status of job -->
                    <xsd:element name="stopStatus"
type="stopJobStatusType" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<!-- dmStart -->
<xsd:element name="dmStart">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="changedJobDefinitionType">
                <xsd:sequence>
                    <xsd:element name="job_security"
minOccurs="0" type="securityType" />
                    <xsd:element
name="command_start_time" minOccurs="0" type="xsd:dateTime" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

```

```

<!-- dmStartOutput -->
<xsd:element name="dmStartOutput">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="dmOutputBase">
        <xsd:sequence>
          <!-- job_name, null if pending in queue -->
          <xsd:element name="job_name"
minOccurs="0" type="xsd:string" />

          <!-- jobExecution, null if pending in queue -->
          <xsd:element
name="job_execution_ID" minOccurs="0"
                                type="xsd:long" />
          <!-- status of job -->
          <xsd:element name="startStatus"
type="startJobStatusType" />

          <!-- Any message to Print in addition to standard
message -->
          <xsd:element name="messageCode"
minOccurs="0" type="xsd:string" />

          <!-- Any objects names to send to command line --
>
          <xsd:element name="objectList" minOccurs="0"
maxOccurs="unbounded" type="xsd:string" />

        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<!-- Valid Selection values for objects -->
<xsd:simpleType name="jobVariablesChangeType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="UNCHANGED" />
    <xsd:enumeration value="PARTIAL" />
    <xsd:enumeration value="FULL" />
  </xsd:restriction>
</xsd:simpleType>

```



```

    <!-- dmSync -->
    <xsd:element name="dmSync">
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="dmCommandBase">
                    <xsd:sequence>
                        <!-- job_name, 12315DFHJKS, Job
ID of the job to be started -->
                        <xsd:element name="job_name"
nillable="false"
                                type="xsd:string" />
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>

    <!-- dmSyncOutput -->
    <xsd:element name="dmSyncOutput">
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="dmOutputBase">
                    <xsd:sequence>
                        <!-- job_name, null if pending in queue -->
                        <xsd:element name="job_name"
minOccurs="0" type="xsd:string" />

                        <!-- jobExecution, null if pending in queue -->
                        <xsd:element
name="job_execution_ID" minOccurs="0"
                                type="xsd:long" />
                        <!-- status of job -->
                        <xsd:element name="status"
type="statusType" />

                        <!-- row count validation errors -->
                        <xsd:element
name="rowCountErrors" type="xsd:string"
                                maxOccurs="unbounded"
minOccurs="0" />
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>

```

```

<!-- dmCleanup -->
<xsd:element name="dmCleanup">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="dmCommandBase">
        <xsd:sequence>
          <!-- job_name, 12315DFHJKS, Job ID of the job to
be started -->
          <xsd:element name="job_name"
nillable="false" type="xsd:string" />
          <xsd:element
name="response_timeout" minOccurs="0"
type="xsd:int"
default="10" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<!-- dmCleanupOutput -->
<xsd:element name="dmCleanupOutput">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="dmOutputBase">
        <xsd:sequence>
          <xsd:element name="job_name"
nillable="false" type="xsd:string" />
          <!-- status of cleanup job -->
          <xsd:element name="message"
type="xsd:string" />
          <xsd:element name="result"
type="xsd:string"
maxOccurs="unbounded"
minOccurs="0" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

  <!-- dmUpdateJobSteps -->
  <xsd:element name="dmUpdateJobSteps">

```

```

        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="dmCommandBase">
                    <xsd:sequence>
                        <!-- job_name, 12315DFHJKS, Job ID of the job to
be started -->
                        <xsd:element name="job_name"
nillable="false" type="xsd:string" />
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>

    <!-- dmUpdateJobStepsOutput -->
    <xsd:element name="dmUpdateJobStepsOutput">
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="dmOutputBase">
                    <xsd:sequence>
                        <xsd:element name="job_name"
nillable="false" type="xsd:string" />
                        <!-- status of cleanup job -->
                        <xsd:element name="result"
type="xsd:string"
maxOccurs="unbounded"
minOccurs="0" />
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>

    <!-- dmStatus -->
    <xsd:element name="dmStatus">
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="dmCommandBase">
                    <xsd:sequence>
                        <xsd:choice>
                            <!-- Either job_name or job_id -->
                            <!-- job_name, JOB1, Job Name of the
job to get status -->
                            <xsd:element name="job_name"

```

```

type="xsd:string"/>
                                <!-- job_id, 1321245456, Job ID of the job
to get status -->
                                <xsd:element name="job_id"
type="xsd:long"/>
                                </xsd:choice>
                                <!--
of polling in seconds, negative
                                frequency, 10, Frequency
again
                                indicates do no poll
-->
                                <xsd:element name="frequency"
minOccurs="0" type="xsd:int" />
                                <!--
level 1: Basic job status, 2: Job steps
                                Job status granularity
step status
                                status, 3: Detailed job
-->
                                <xsd:element name="output_level"
minOccurs="0" type="xsd:int"
                                default="1" />
                                <xsd:element
name="response_timeout" minOccurs="0"
                                type="xsd:int"
default="10" />
                                </xsd:sequence>
                                </xsd:extension>
                                </xsd:complexContent>
                                </xsd:complexType>
                                </xsd:element>

                                <!-- output from dmStatus -->
                                <xsd:element name="dmStatusOutput">
                                    <xsd:complexType>
                                        <xsd:complexContent>
                                            <xsd:extension base="dmOutputBase">
                                                <xsd:sequence>
                                                    <!-- job_name, 12315DFHJKS, Job ID of the job to
be started -->
                                                    <xsd:element name="job_name"
nillable="false" type="xsd:string" />

```

```

                                <!-- jobExecution to be copied -->
                                <xsd:element name="job_execution"
nillable="false"
                                type="jobExecutionType" />
                                </xsd:sequence>
                                </xsd:extension>
                                </xsd:complexContent>
                                </xsd:complexType>
                                </xsd:element>

                                <!-- dmViewLog -->
                                <xsd:element name="dmViewLog">
                                    <xsd:complexType>
                                        <xsd:complexContent>
                                            <xsd:extension base="dmCommandBase">
                                                <xsd:sequence>
                                                    <!-- job_name, 12315DFHJKS, Job ID of the job to
be started -->
                                                    <xsd:element name="job_name"
nillable="false" type="xsd:string" />
                                                    <!--
                                                    dir /user/tptapi/lists,
(optional)Output directory for log file.
                                                    -->
                                                    <xsd:element name="dir"
minOccurs="0" type="xsd:string" />
                                                    <!--
                                                    filename: log.txt,
(optional)Output file name for the log file.
                                                    Will be overwritten if
already exist
                                                    -->
                                                    <xsd:element name="filename"
minOccurs="0" type="xsd:string" />
                                                    <xsd:element
name="response_timeout" minOccurs="0"
                                                    type="xsd:int"
default="10" />
                                                </xsd:sequence>
                                            </xsd:extension>
                                        </xsd:complexContent>
                                    </xsd:complexType>
                                </xsd:element>

```

```

    <!-- Valid status values:
         A (ALL) N (NEW) I (INITIALIZING), R (RUNNING), C
(COMPLETED_SUCCESSFULLY), F (FAILED),
         RS(RESTARTING), Q (QUEUED), UC (USER_CANCELLED)
    -->
    <xsd:simpleType name="statusModeType">
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="A|a|N|n|I|i|R|r|B|b|C|c|F|f|[Rr][Ss]|
Q|q|[Uu][Cc]"/>
        </xsd:restriction>
    </xsd:simpleType>

    <!-- DmListJobs -->
    <xsd:element name="dmListJobs">
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="dmCommandBase">
                    <xsd:sequence>
                        <xsd:element name="status_mode"
minOccurs="0" type="statusModeType" default="A"/>
                        <xsd:element name="job_name"
minOccurs="0" type="xsd:string" />
                        <xsd:element
name="job_execution_name" minOccurs="0" type="xsd:string" />
                        <xsd:element name="start_time"
minOccurs="0" type="xsd:string" />
                        <xsd:element name="end_time"
minOccurs="0" type="xsd:string" />
                        <xsd:element
name="end_time_after" minOccurs="0" type="xsd:string" />
                        <xsd:element name="last_hour"
minOccurs="0" type="xsd:int" default="0" />
                        <xsd:element
name="freeze_step_only" minOccurs="0" type="xsd:boolean" default="false" />
                        <xsd:element
name="latest_job_only" minOccurs="0" type="xsd:boolean" default="true" />
                        <xsd:element
name="response_timeout" minOccurs="0"
                                type="xsd:int"
                                default="10" />
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>

```

```

        </xsd:complexType>
    </xsd:element>

    <!-- DmListJobsOutput -->
    <xsd:element name="dmListJobsOutput">
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="dmOutputBase">
                    <xsd:sequence>
                        <xsd:element
name="listJobsRecord" type="listJobsRecordType"
                                maxOccurs="unbounded"
minOccurs="0" />
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>

    <!-- Valid status values: A (ALL), R (RUNNING), Q (QUEUED) -->
    <xsd:simpleType name="taskStatusModeType">
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="A|a|R|r|Q|q" />
        </xsd:restriction>
    </xsd:simpleType>

    <!-- listTasksRecordType -->
    <xsd:complexType name="listTasksRecordType">
        <xsd:sequence>
            <xsd:element name="task_id" type="xsd:long" />
            <xsd:element name="job_name" minOccurs="0"
type="xsd:string" />
            <xsd:element name="task_status" type="xsd:string" />
            <xsd:element name="agent_names" minOccurs="0"
maxOccurs="unbounded" type="xsd:string" />
            <xsd:element name="queue_name" minOccurs="0"
type="xsd:string" />
            <xsd:element name="queue_order" type="xsd:int"
nillable="true"/>
            <xsd:element name="utility" type="xsd:string" />
            <xsd:element name="src_system" minOccurs="0"
type="xsd:string" />
            <xsd:element name="target_system" minOccurs="0"
type="xsd:string" />

```

```

        <xsd:element name="target_user_pool_name" minOccurs="0"
type="xsd:string" />
        <xsd:element name="job_priority" type="xsd:string" />
        <xsd:element name="last_update_time" type="xsd:long" />
    </xsd:sequence>
</xsd:complexType>

<!-- DmListTasks -->
<xsd:element name="dmListTasks">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmCommandBase">
                <xsd:sequence>
                    <xsd:element
name="task_status_mode" minOccurs="0"
                                type="taskStatusModeType"
default="A" />
                    <xsd:element name="job_name"
minOccurs="0" type="xsd:string" />
                    <xsd:element name="task_id"
minOccurs="0"
                                type="xsd:long" />
                    <xsd:element name="agent_name"
minOccurs="0" type="xsd:string" />
                    <xsd:element name="frequency"
minOccurs="0" type="xsd:int" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<!-- DmListTasksOutput -->
<xsd:element name="dmListTasksOutput">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmOutputBase">
                <xsd:sequence>
                    <xsd:element
name="listTasksRecordTypeList" type="listTasksRecordType"
                                maxOccurs="unbounded"
minOccurs="0" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

```



```

        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<xsd:element name="dmTaskListUpdateAnnouncement">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmMessageBase">
                <xsd:sequence>
                    <xsd:element name="timestamp"
type="xsd:long" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<!-- Get Datamover API Info -->
<xsd:element name="dmGetAPIInfo">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmCommandBase">
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>

<!-- Datamover API INFO -->
<xsd:element name="dmAPIInfoOutput">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmOutputBase">
                <xsd:sequence>
                    <xsd:element name="version"
nillable="false" type="xsd:string" />
                    <xsd:element name="time"
nillable="false" type="xsd:string" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<!-- DSA target groups -->

```

```

    <xsd:element name="dmDSATargetGroups">
      <xsd:complexType>
        <xsd:complexContent>
          <xsd:extension base="dmCommandBase">
            </xsd:extension>
          </xsd:complexContent>
        </xsd:complexType>
      </xsd:element>

    <!-- DSA target groups output -->
    <xsd:element name="dmDSATargetGroupsOutput">
      <xsd:complexType>
        <xsd:complexContent>
          <xsd:extension base="dmOutputBase">
            <xsd:sequence>
              <xsd:element name="targetGroups"
minOccurs="0" maxOccurs="unbounded"
                                type="xsd:string" />
            </xsd:sequence>
          </xsd:extension>
        </xsd:complexContent>
      </xsd:complexType>
    </xsd:element>

    <xsd:complexType name="jobStatusNotificationType">
      <xsd:sequence>
        <xsd:element name="jobName" minOccurs="1" type="xsd:string" />
      </xsd:sequence>
    </xsd:complexType>

    <!-- news push type -->
    <xsd:complexType name="newsPushType">
      <xsd:complexContent>
        <xsd:extension base="jobStatusNotificationType">
          <xsd:sequence>
            <xsd:element name="channel" minOccurs="1"
type="xsd:string" />
            <xsd:element name="timestamp" minOccurs="1"
type="xsd:long" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>

```

```

<!-- DmNewPush -->
<xsd:element name="dmNewsPush">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="dmCommandBase">
        <xsd:sequence>
          <xsd:element name="newsPush"
minOccurs="1" type="newsPushType" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<!-- DMNewsPush output -->
<xsd:element name="dmNewsPushOutput">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="dmOutputBase"/>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<!-- dsaJobStatus element of Cloud staging news push type -->
<xsd:complexType name="cs2DsaJobStatusType">
  <xsd:sequence>
    <xsd:element name="job_name" minOccurs="1" type="xsd:string" />
    <xsd:element name="job_type" minOccurs="1" type="xsd:string" />
    <xsd:element name="job_status" minOccurs="1" type="xsd:string" />
    <xsd:element name="percent_complete" minOccurs="1" type="xsd:int" />
    <xsd:element name="elapsed_time" minOccurs="1" type="xsd:string" />
  </xsd:sequence>
</xsd:complexType>

<!-- Cloud Staging news push type -->
<xsd:complexType name="cloudStagingNewsPushType">
  <xsd:complexContent>
    <xsd:extension base="jobStatusNotificationType">
      <xsd:sequence>
        <xsd:element name="channel" minOccurs="1"
type="xsd:string" />
        <xsd:element name="job_status" minOccurs="0"
type="xsd:string" />
        <xsd:element name="timestamp" minOccurs="0"

```

```

type="xsd:long" />
        <xsd:element name="elapsed_time" minOccurs="0"
type="xsd:string" />
        <xsd:element name="dsa_job_status" minOccurs="0"
type="cs2DsaJobStatusType" />
    </xsd:sequence>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<!-- dmCsNewsPush -->
<xsd:element name="dmCs2NewsPush">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmCommandBase">
                <xsd:sequence>
                    <xsd:element name="newsPush"
minOccurs="1" type="cloudStagingNewsPushType" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<!-- dmDeleteJob -->
<xsd:element name="dmDeleteJob">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmCommandBase">
                <xsd:sequence>
                    <!-- job_name, 12315DFHJKS, Job ID of the job to
be started -->
                    <xsd:element name="job_name"
nillable="false" type="xsd:string" />
                    <!-- Delete all of job definition and job history
of this job -->
                    <xsd:element name="all"
minOccurs="0" type="booleanType" />
                    <!-- Only one is allowed depending if using
Command Line or JMS directly -->
                    <xsd:choice>
                        <!-- Not used when using Command Line.
Preview the list of job that will be deleted -->
                        <xsd:element

```

```

name="preview" minOccurs="0" type="xsd:boolean" />
        <!-- Only used by Command Line. Skip
preview and confirmation and delete directly -->
        <xsd:element
name="skip_prompt" minOccurs="0" type="booleanType" />
        </xsd:choice>
        <xsd:element
name="response_timeout" minOccurs="0"
        type="xsd:int"
default="10" />
        </xsd:sequence>
    </xsd:extension>
</xsd:complexContent>
</xsd:complexType>
</xsd:element>

<!-- dmDeleteJobOutput -->
<xsd:element name="dmDeleteJobOutput">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmOutputBase">
                <xsd:sequence>
                    <xsd:element name="job"
minOccurs="0"
                    maxOccurs="unbounded"
type="simpleJobType" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<!-- DMGetJobDefinition -->
<xsd:element name="dmGetJobDefinition">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmCommandBase">
                <xsd:sequence>
                    <!-- job_name, 12315DFHJKS, Job Name of the job
definition -->
                    <xsd:element name="job_name"
nillable="false" type="xsd:string" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

```

```

(optional)Output directory for object
list.
-->
<xsd:element name="dir"
minOccurs="0" type="xsd:string" />
<!--
filename
jobdefinition.xml, (optional)Output file name for the
job definition file. Will
be overwritten if already exist
-->
<xsd:element name="filename"
minOccurs="0" type="xsd:string" />
<xsd:element name="public_key"
minOccurs="0" type="xsd:string" />
<xsd:element
name="response_timeout" minOccurs="0"
type="xsd:int"
default="10" />
</xsd:sequence>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>
</xsd:element>

<!-- dmGetJobDefinitionOutput -->
<xsd:element name="dmGetJobDefinitionOutput">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="dmOutputBase">
        <xsd:sequence>
          <xsd:element
name="job_definition" type="jobDefinitionType" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<!-- dmListJobStep: Get job steps -->
<xsd:element name="dmListJobSteps">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="dmCommandBase">

```

```

                                <xsd:sequence>
                                    <xsd:element name="job_name"
nillable="false" type="xsd:string" />
                                    <xsd:element name="dir"
minOccurs="0" type="xsd:string" />
                                    <xsd:element name="filename"
minOccurs="0" type="xsd:string" />
                                    <xsd:element
name="response_timeout" minOccurs="0"
                                type="xsd:int"
                                default="10" />
                                </xsd:sequence>
                            </xsd:extension>
                        </xsd:complexContent>
                    </xsd:complexType>
                </xsd:element>

                <!-- dmLisJobStepsOutput -->
                <xsd:element name="dmListJobStepsOutput">
                    <xsd:complexType>
                        <xsd:complexContent>
                            <xsd:extension base="dmOutputBase">
                                <xsd:sequence>
                                    <xsd:element name="job_name"
type="xsd:string" />
                                    <xsd:element name="source_tdpid"
minOccurs="0" type="xsd:string" />
                                    <xsd:element name="source_user"
minOccurs="0" type="xsd:string" />
                                    <xsd:element
name="source_aster_login" minOccurs="0" type="asterSystemType"/>
                                    <xsd:element name="target_tdpid"
minOccurs="0" type="xsd:string" />
                                    <xsd:element name="target_user"
minOccurs="0" type="xsd:string" />
                                    <xsd:element
name="target_aster_login" minOccurs="0" type="asterSystemType"/>
                                    <xsd:element name="step" minOccurs="0"
maxOccurs="unbounded" type="stepListType" />
                                    <xsd:element
name="source_sessions" minOccurs="0" type="sessionsAndStreamsType" />
                                    <xsd:element
name="target_sessions" minOccurs="0" type="sessionsAndStreamsType" />
                                    <xsd:element name="data_streams"

```

```

minOccurs="0" type="sessionsAndStreamsType" />
        </xsd:sequence>
    </xsd:extension>
</xsd:complexContent>
</xsd:complexType>
</xsd:element>

<!-- Get Daemon Public Key -->
<xsd:element name="dmCryptoPublicKey">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmCommandBase">
                <xsd:sequence>
                    <xsd:element
name="response_timeout" minOccurs="0"
                                type="xsd:int" default="10" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<xsd:element name="dmCryptoPublicKeyOutput">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmOutputBase">
                <xsd:sequence>
                    <xsd:element name="public_key"
type="xsd:string"
                                minOccurs="1"
maxOccurs="1" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<!-- Request Encrypted Password Command -->
<xsd:element name="dmGetEncryptedPassword">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmCommandBase">
                <xsd:sequence>
                    <xsd:element name="password"

```



```

type="xsd:string" />
nillable="false"
<!--
dir /user/tptapi/lists,
list.
-->
<xsd:element name="dir"
minOccurs="0" type="xsd:string" />
<!--
object_list
object list
-->
<xsd:element name="filename"
<xsd:element
name="response_timeout" minOccurs="0"
type="xsd:int" default="10" />
</xsd:sequence>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>
</xsd:element>
<xsd:element name="dmGetEncryptedPasswordOutput">
<xsd:complexType>
<xsd:complexContent>
<xsd:extension base="dmOutputBase">
<xsd:sequence>
<xsd:element
name="password_encrypted" type="xsd:string"
minOccurs="1"
maxOccurs="1" />
</xsd:sequence>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>
</xsd:element>
<!-- stepListType -->
<xsd:complexType name="stepListType">
<xsd:sequence>
<xsd:element name="data_streams" minOccurs="0" maxOccurs="1"
type="xsd:int"/>

```

```

        <xsd:element name="dsa_task" minOccurs="0"
maxOccurs="unbounded" type="DSATaskType" />
        <xsd:element name="arc_task" minOccurs="0"
maxOccurs="unbounded" type="ARCTaskType" />
        <xsd:element name="jdbc_task" minOccurs="0"
maxOccurs="unbounded" type="JDBCTaskType" />
        <xsd:element name="sql_task" minOccurs="0"
maxOccurs="unbounded" type="SQLTaskType" />
        <xsd:element name="aster_mr_task" minOccurs="0"
maxOccurs="unbounded" type="AsterMRTaskType" />
        <xsd:element name="tptapi_task" minOccurs="0"
maxOccurs="unbounded" type="TPTAPITaskType" />
        <xsd:element name="t2t_task" minOccurs="0"
maxOccurs="unbounded" type="T2TTaskType" />
        <xsd:element name="cs2_task" minOccurs="0" maxOccurs="unbounded"
type="CS2TaskType" />
        <xsd:element name="rowcountvalidation_task" minOccurs="0"
maxOccurs="unbounded" type="RowCountValidationTaskType" />
    </xsd:sequence>
    <xsd:attribute name="type" type="xsd:string" use="required" />
    <xsd:attribute name="id" type="xsd:long" use="required" />
</xsd:complexType>

<xsd:complexType name="DSATaskType">
    <xsd:sequence>
        <xsd:element name="dsa_job_name" type="xsd:string" minOccurs="0"
maxOccurs="1" />
        <xsd:element name="job_model_json" type="xsd:string" minOccurs="0"
maxOccurs="1"/>
        <xsd:element name="data_streams" type="xsd:int" minOccurs="0"
maxOccurs="1" default="1" />
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:long" use="required" />
</xsd:complexType>

<xsd:complexType name="CS2TaskType">
    <xsd:sequence>
        <xsd:element name="cs2_job_name" type="xsd:string" minOccurs="0"
maxOccurs="1" />
        <xsd:element name="job_model_json" type="xsd:string" minOccurs="0"
maxOccurs="1"/>
        <xsd:element name="data_streams" type="xsd:int" minOccurs="0"
maxOccurs="1" default="1" />
    </xsd:sequence>

```

```

    <xsd:attribute name="id" type="xsd:long" use="required" />
  </xsd:complexType>

  <xsd:complexType name="ARCTaskType">
    <xsd:sequence>
      <xsd:element name="source_unlock_script"
type="xsd:string" minOccurs="0" maxOccurs="1" />
      <xsd:element name="target_unlock_script"
type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="archive_script" type="xsd:string"
minOccurs="0" maxOccurs="1"/>
      <xsd:element name="copy_script" type="xsd:string"
minOccurs="0" maxOccurs="1"/>
      <xsd:element name="build_script" type="xsd:string"
minOccurs="0" maxOccurs="1"/>
      <xsd:element name="source_sessions_count" type="xsd:int"
default="1"/>
      <xsd:element name="target_sessions_count" type="xsd:int"
default="1"/>
      <xsd:element name="data_streams" type="xsd:int"
minOccurs="0" maxOccurs="1" default="1" />
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:long" use="required" />
  </xsd:complexType>

  <xsd:complexType name="SQLTaskType">
    <xsd:sequence>
      <xsd:element name="object_move_phrase" type="xsd:string"
minOccurs="0" />
      <xsd:element name="source" type="objectInfoType"
minOccurs="0" maxOccurs="1"/>
      <xsd:element name="target" type="objectInfoType"
minOccurs="0" maxOccurs="1"/>
      <xsd:element name="data_streams" type="xsd:int"
minOccurs="0" maxOccurs="1" default="1" />
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:long" use="required" />
  </xsd:complexType>

  <xsd:complexType name="AsterMRTaskType">
    <xsd:sequence>
      <xsd:element name="object_type" type="xsd:string" />
      <xsd:element name="object_move_phrase" type="xsd:string"
minOccurs="0" />

```

```

        <xsd:element name="movement_type" type="xsd:string"
minOccurs="0" />
        <xsd:element name="source" type="asterObjectInfoType" />
        <xsd:element name="target" type="asterObjectInfoType" />
        <xsd:element name="aster_sql_mr_statement"
type="xsd:string" minOccurs="0" />
        <xsd:element name="data_streams" type="xsd:int"
minOccurs="0" maxOccurs="1" default="1" />
        <!--
        <xsd:element name="aster_progress_query"
type="xsd:string" minOccurs="0" />
        <xsd:element name="teradata_progress_query"
type="xsd:string" minOccurs="0" />
        -->
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:long" use="required" />
</xsd:complexType>

<xsd:complexType name="TPTAPITaskType">
    <xsd:sequence>
        <xsd:element name="object_type" type="xsd:string" />
        <xsd:element name="source_sessions_count" type="xsd:int"
default="1" />
        <xsd:element name="target_sessions_count" type="xsd:int"
default="1" />
        <xsd:element name="source" type="objectInfoType" />
        <xsd:element name="target" type="objectInfoType" />
        <xsd:element name="data_streams" type="xsd:int"
minOccurs="0" maxOccurs="1" default="1" />
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:long" use="required" />
</xsd:complexType>

<xsd:complexType name="T2TTaskType">
    <xsd:sequence>
        <xsd:element name="object_type" type="xsd:string" />
        <xsd:element name="source" type="objectInfoType" />
        <xsd:element name="target" type="objectInfoType" />
        <xsd:element name="data_streams" type="xsd:int"
minOccurs="0" maxOccurs="1" default="1" />
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:long" use="required" />
</xsd:complexType>

```

```

<xsd:complexType name="JDBCTaskType">
  <xsd:sequence>
    <xsd:element name="object_type" type="xsd:string" />
    <xsd:element name="source" type="objectInfoType" />
    <xsd:element name="target" type="objectInfoType" />
    <xsd:element name="data_streams" type="xsd:int"
minOccurs="0" maxOccurs="1" default="1" />
  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:long" use="required" />
</xsd:complexType>

<xsd:complexType name="RowCountValidationTaskType">
  <xsd:sequence>
    <xsd:element name="object_type" type="xsd:string" />
    <xsd:element name="source" type="objectInfoType" />
    <xsd:element name="target" type="objectInfoType" />
    <xsd:element name="data_streams" type="xsd:int"
minOccurs="0" maxOccurs="1" default="1" />
  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:long" use="required" />
</xsd:complexType>

<xsd:complexType name="objectInfoType">
  <xsd:sequence>
    <xsd:element name="parent" type="xsd:string"
minOccurs="0" />
    <xsd:element name="table" type="xsd:string"
minOccurs="0" />
    <xsd:element name="operator_type" type="xsd:string"
minOccurs="0" />
    <xsd:element name="sql_statement_list">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="sql_statement"
type="xsd:string"
minOccurs="0"
maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <!--
      <xsd:element name="sql_statement_list"
type="sqlStatementType"
minOccurs="0" />
    </xsd:element

```

```

name="sql_statement_list"
                                type="sqlStatementType" minOccurs="0"
maxOccurs="unbounded" />
                                -->
                                </xsd:sequence>
                                </xsd:complexType>

                                <xsd:complexType name="PostLoadSQLType">
                                    <xsd:sequence>
                                        <xsd:element name="sql" type="xsd:string"
minOccurs="0" />
                                        <xsd:element name="startTime" type="xsd:dateTime"
minOccurs="0"/>
                                        <xsd:element name="stopTime" type="xsd:dateTime"
minOccurs="0"/>
                                    </xsd:sequence>
                                </xsd:complexType>

                                <xsd:complexType name="asterObjectInfoType">
                                    <xsd:sequence>
                                        <xsd:element name="parent" type="xsd:string"
minOccurs="0" />
                                        <xsd:element name="schema" type="xsd:string"
minOccurs="0" />
                                        <xsd:element name="table" type="xsd:string"
minOccurs="0" />
                                    </xsd:sequence>
                                </xsd:complexType>

                                <xsd:complexType name="eventTableType">
                                    <xsd:sequence>
                                        <xsd:element name="event_table_name" nillable="false"
type="xsd:string" />
                                        <xsd:element name="system" nillable="false"
type="xsd:string" />
                                        <xsd:element name="database" nillable="false"
type="xsd:string" />
                                        <xsd:element name="user_name" nillable="false"
type="xsd:string" />
                                        <xsd:element name="user_password" nillable="false"
type="xsd:string" />
                                        <xsd:element name="event_table_id" nillable="false"
type="xsd:long" />
                                    </xsd:sequence>

```

```

</xsd:complexType>

<!-- dmEventTableBase -->
<xsd:complexType name="dmEventTableBase">
  <xsd:complexContent>
    <xsd:extension base="dmCommandBase">
      <xsd:sequence>
        <!--
                                name for this installation of
                                event table.
        -->
        <xsd:element name="event_table_name"
nillable="false"
                                type="xsd:string" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<!-- dmCreateTMSMEventTable -->
<xsd:element name="dmCreateEventTable">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="dmEventTableBase">
        <xsd:sequence>
          <!--
                                system - Teradata System
where TMSMEvent Table Resides
          -->
          <xsd:element name="system"
nillable="false"
                                type="xsd:string" />
          <!--
                                user_name - Teradata user
name used to access TMSMEvent Table
          -->
          <xsd:element name="user_name"
nillable="false"
                                type="xsd:string" />
          <!--
                                user_password (optional)
          -->
          <xsd:element name="user_password"
minOccurs="0"
                                type="xsd:string" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

```

```

                                type="xsd:string" />
                                <!-- internal use only -->
                                <xsd:element
name="user_password_encrypted"
                                minOccurs="0"
                                type="xsd:base64Binary" />
                                <!--
                                Parent Database for
TMSMEvent Table
                                -->
                                <xsd:element
name="event_database" nillable="false"
                                type="xsd:string" />
                                <!--
                                Using existing TSM Event
Table (optional)
                                -->
                                <xsd:element
name="use_existing_event_table"
                                minOccurs="0"
                                type="booleanType" />
                                </xsd:sequence>
                                </xsd:extension>
                                </xsd:complexContent>
                                </xsd:complexType>
                                </xsd:element>

                                <!-- dmCreateEventTableOutput -->
                                <xsd:element name="dmCreateEventTableOutput">
                                <xsd:complexType>
                                <xsd:complexContent>
                                <xsd:extension base="dmOutputBase">
                                <xsd:sequence>
                                <!-- name of the tsm event table
created -->
                                <xsd:element
name="event_table_name" nillable="false"
                                type="xsd:string" />
                                <xsd:element name="create_result"
                                maxOccurs="unbounded"
                                minOccurs="0"></xsd:element>

```



```

        </xsd:sequence>
    </xsd:extension>
</xsd:complexContent>
</xsd:complexType>
</xsd:element>

<!-- dmDeleteEventTable -->
<xsd:element name="dmDeleteEventTable">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmEventTableBase">
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<!-- dmDeleteEventTableOutput -->
<xsd:element name="dmDeleteEventTableOutput">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmOutputBase">
                <xsd:sequence>
                    <xsd:element
name="event_table_name" nillable="false"
                                type="xsd:string" />
                    <xsd:element name="result"
type="xsd:string" maxOccurs="unbounded" minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<!-- dmModifyEventTable -->
<xsd:element name="dmModifyEventTable">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmEventTableBase">
                <xsd:sequence>
                    <!-- Change user_name -->
                    <xsd:element
name="change_user_name" minOccurs="0"
                                type="booleanType"
default="false"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

```

```

minOccurs="0" type="xsd:string" />

name="change_user_password" minOccurs="0"

default="false"/>

minOccurs="0" type="xsd:string" />

name="user_password_encrypted" minOccurs="0"

type="xsd:base64Binary" />

</xsd:sequence>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>
</xsd:element>

<!-- dmEventTableOutput -->
<xsd:element name="dmEventTableOutput">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="dmOutputBase">
        <xsd:sequence>
          <xsd:element
name="event_table_name" nillable="false"
                                type="xsd:string" />
          <xsd:element name="result"
type="xsd:string" maxOccurs="unbounded" minOccurs="0"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<!-- dmListEventTable -->
<xsd:element name="dmListEventTable">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="dmCommandBase">

```

```

        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
</xsd:element>

<xsd:element name="dmListEventTableOutput">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmOutputBase">
                <xsd:sequence>
                    <xsd:element name="result"
type="eventTableType" minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<!-- dmListConfiguration -->
<xsd:element name="dmListConfiguration">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmCommandBase">
                <xsd:sequence>
                    <!--
                                dir /user/tptapi/lists,
(optional)Output directory for object
                                list.
                    -->
                    <xsd:element name="dir"
minOccurs="0" type="xsd:string" />
                    <!--
                                filename
jobdefinition.xml, (optional)Output file name for the
                                job definition file. Will
be overwritten if already exist
                                -->
                    <xsd:element name="filename"
minOccurs="0" type="xsd:string" />
                    <xsd:element
name="response_timeout" minOccurs="0"
                                type="xsd:int"
default="10" />
                    <xsd:element name="public_key"

```

```

minOccurs="0" type="xsd:string" />
        </xsd:sequence>
    </xsd:extension>
</xsd:complexContent>
</xsd:complexType>
</xsd:element>

<!-- Output for dmListConfiguration -->
<xsd:element name="configurationOutput">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmOutputBase">
                <xsd:sequence>
                    <xsd:element name="property"
nillable="false" maxOccurs="unbounded" type="propertyType" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<!-- dmSaveConfiguration -->
<xsd:element name="dmSaveConfiguration">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmCommandBase">
                <xsd:sequence>
                    <xsd:element name="property"
nillable="false" maxOccurs="unbounded" type="propertyType" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<!-- Output for dmSaveConfiguration -->
<xsd:element name="saveConfigurationOutput">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmOutputBase">
                <xsd:sequence>
                    <xsd:element
name="updated_property" nillable="false" minOccurs="0" maxOccurs="unbounded"
type="propertyType" />

```

```

                                <xsd:element
name="targetUserProperty" nillable="true" type="propertyType" />
                                <xsd:element
name="groupUserProperty" nillable="true" type="propertyType" />
                                </xsd:sequence>
                            </xsd:extension>
                        </xsd:complexContent>
                    </xsd:complexType>
                </xsd:element>

                <!-- dmListAgents -->
                <xsd:element name="dmListAgents">
                    <xsd:complexType>
                        <xsd:complexContent>
                            <xsd:extension base="dmCommandBase">
                                <xsd:sequence>
                                    <xsd:element name="agent_names"
minOccurs="0" type="xsd:string" />
                                </xsd:sequence>
                            </xsd:extension>
                        </xsd:complexContent>
                    </xsd:complexType>
                </xsd:element>

                <!-- Output for dmAgentListOutput -->
                <xsd:element name="dmAgentListOutput">
                    <xsd:complexType>
                        <xsd:complexContent>
                            <xsd:extension base="dmOutputBase">
                                <xsd:sequence>
                                    <xsd:element name="agent_info"
minOccurs="0" maxOccurs="unbounded" type="agentInfoType" />
                                </xsd:sequence>
                            </xsd:extension>
                        </xsd:complexContent>
                    </xsd:complexType>
                </xsd:element>

                <!-- DMPingDaemon -->
                <xsd:element name="dmPingDaemon">
                    <xsd:complexType>
                        <xsd:complexContent>
                            <xsd:extension base="dmCommandBase">
                                <xsd:sequence>

```

```

                                <!-- job_name, 12315DFHJKS, Job Name of the job
definition -->
                                <xsd:element
name="correlation_id" nillable="false"
                                type="xsd:long" />
                                </xsd:sequence>
                                </xsd:extension>
                                </xsd:complexContent>
                                </xsd:complexType>
                                </xsd:element>

                                <!-- getGlobalAccessPermissions -->
                                <xsd:element name="getGlobalAccessPermissions">
                                    <xsd:complexType>
                                        <xsd:complexContent>
                                            <xsd:extension base="dmCommandBase">
                                                </xsd:extension>
                                            </xsd:complexContent>
                                        </xsd:complexType>
                                    </xsd:element>

                                <xsd:element name="getGlobalAccessPermissionsOutput">
                                    <xsd:complexType>
                                        <xsd:complexContent>
                                            <xsd:extension base="dmOutputBase">
                                                <xsd:sequence>
                                                    <xsd:element name="permission"
minOccurs="0" maxOccurs="unbounded" type="accessPermissionType" />
                                                    </xsd:sequence>
                                                </xsd:extension>
                                            </xsd:complexContent>
                                        </xsd:complexType>
                                    </xsd:element>

                                <!-- updateGlobalAccessPermissions -->
                                <xsd:element name="updateGlobalAccessPermissions">
                                    <xsd:complexType>
                                        <xsd:complexContent>
                                            <xsd:extension base="dmCommandBase">
                                                <xsd:sequence>
                                                    <xsd:element name="permission"
minOccurs="0" maxOccurs="unbounded" type="accessPermissionType" />
                                                    </xsd:sequence>
                                                </xsd:extension>
                                            </xsd:complexContent>
                                        </xsd:complexType>
                                    </xsd:element>

```

```

        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
</xsd:element>

<xsd:element name="updateGlobalAccessPermissionsOutput">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmOutputBase">
                <xsd:sequence>
                    <xsd:element name="status"
maxOccurs="1" type="xsd:int" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<!-- AccessPermissionType -->
<xsd:complexType name="accessPermissionType">
    <xsd:sequence>
        <xsd:element name="name" minOccurs="1" maxOccurs="1"
type="xsd:string" />
        <xsd:element name="object_type" minOccurs="1"
maxOccurs="1" type="permissionObjectType" />
        <xsd:element name="allowRead" minOccurs="0" maxOccurs="1"
type="booleanType" />
        <xsd:element name="allowWrite" minOccurs="0"
maxOccurs="1" type="booleanType" />
        <xsd:element name="allowExecute" minOccurs="0"
maxOccurs="1" type="booleanType" />
    </xsd:sequence>
</xsd:complexType>

<!-- ModificationPermissionType -->
<xsd:complexType name="modificationPermissionType">
    <xsd:sequence>
        <xsd:element name="name" minOccurs="1" maxOccurs="1"
type="xsd:string" />
        <xsd:element name="object_type" minOccurs="1"
maxOccurs="1" type="permissionObjectType" />
        <xsd:element name="allowUpdateJobPlan" minOccurs="0"
maxOccurs="1" type="booleanType" />
        <xsd:element name="allowChangeJobPriority" minOccurs="0"

```

```

maxOccurs="1" type="booleanType" />
        <xsd:element name="maxNumberOfStreams" minOccurs="0"
maxOccurs="1" type="xsd:int" />
        <xsd:element name="allowDSA" minOccurs="0" maxOccurs="1"
type="booleanType" />
        <xsd:element name="allowT2T" minOccurs="0" maxOccurs="1"
type="booleanType" />
        <xsd:element name="allowARC" minOccurs="0" maxOccurs="1"
type="booleanType" />
        <xsd:element name="allowTPTAPILOAD" minOccurs="0"
maxOccurs="1" type="booleanType" />
        <xsd:element name="allowTPTAPIUPDATE" minOccurs="0"
maxOccurs="1" type="booleanType" />
        <xsd:element name="allowTPTAPISTREAM" minOccurs="0"
maxOccurs="1" type="booleanType" />
    </xsd:sequence>
</xsd:complexType>

<!-- PermissionObjectType -->
<xsd:simpleType name="permissionObjectType">
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="[Uu][Ss][Ee][Rr]|[Rr][Oo][Ll][Ee]"/>
    </xsd:restriction>
</xsd:simpleType>

<!-- JobContextPermission -->
<xsd:complexType name="jobContextPermission">
    <xsd:sequence>
        <xsd:element name="allowRead" minOccurs="1" maxOccurs="1"
type="xsd:boolean" default="false" />
        <xsd:element name="allowExecute" minOccurs="1"
maxOccurs="1" type="xsd:boolean" default="false" />
        <xsd:element name="allowWrite" minOccurs="1"
maxOccurs="1" type="xsd:boolean" default="false" />
        <xsd:element name="isOwner" minOccurs="1" maxOccurs="1"
type="xsd:boolean" default="false" />
        <xsd:element name="allowUpdateJobPlan" minOccurs="1"
maxOccurs="1" type="xsd:boolean" default="true"/>
        <xsd:element name="allowChangeJobPriority" minOccurs="1"
maxOccurs="1" type="xsd:boolean" default="true"/>
        <xsd:element name="isJobFrozen" minOccurs="1"
maxOccurs="1" type="triStateType" default="unspecified" />
    </xsd:sequence>
</xsd:complexType>

```



```

        </xsd:sequence>
    </xsd:complexType>

    <!-- getGlobalModificationPermissions -->
    <xsd:element name="getGlobalModificationPermissions">
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="dmCommandBase">
                    </xsd:extension>
                </xsd:complexContent>
            </xsd:complexType>
        </xsd:element>

        <xsd:element name="getGlobalModificationPermissionsOutput">
            <xsd:complexType>
                <xsd:complexContent>
                    <xsd:extension base="dmOutputBase">
                        <xsd:sequence>
                            <xsd:element name="permission"
minOccurs="0" maxOccurs="unbounded" type="modificationPermissionType" />
                        </xsd:sequence>
                    </xsd:extension>
                </xsd:complexContent>
            </xsd:complexType>
        </xsd:element>

    <!-- updateGlobalModificationPermissions -->
    <xsd:element name="updateGlobalModificationPermissions">
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="dmCommandBase">
                    <xsd:sequence>
                        <xsd:element name="permission"
minOccurs="0" maxOccurs="unbounded" type="modificationPermissionType" />
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="updateGlobalModificationPermissionsOutput">
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="dmOutputBase">

```

```

                                <xsd:sequence>
                                    <xsd:element name="status"
maxOccurs="1" type="xsd:int" />
                                </xsd:sequence>
                            </xsd:extension>
                        </xsd:complexContent>
                    </xsd:complexType>
                </xsd:element>

                <xsd:element name="getUserPermissionOnJob">
                    <xsd:complexType>
                        <xsd:complexContent>
                            <xsd:extension base="dmCommandBase">
                                <xsd:sequence>
                                    <xsd:element name="job_name"
minOccurs="1" type="xsd:string" />
                                </xsd:sequence>
                            </xsd:extension>
                        </xsd:complexContent>
                    </xsd:complexType>
                </xsd:element>

                <xsd:complexType name="jobAccessPermissionType">
                    <xsd:sequence>
                        <xsd:element name="isOwner" minOccurs="0" maxOccurs="1"
type="booleanType" />
                        <xsd:element name="allowRead" minOccurs="0" maxOccurs="1"
type="booleanType" />
                        <xsd:element name="allowWrite" minOccurs="0"
maxOccurs="1" type="booleanType" />
                        <xsd:element name="allowExecute" minOccurs="0"
maxOccurs="1" type="booleanType" />
                    </xsd:sequence>
                </xsd:complexType>

                <xsd:element name="getUserPermissionOnJobOutput">
                    <xsd:complexType>
                        <xsd:complexContent>
                            <xsd:extension base="dmOutputBase">
                                <xsd:sequence>
                                    <xsd:element
name="job_access_permission" minOccurs="1" maxOccurs="1"
type="jobAccessPermissionType" />

```

```

        </xsd:sequence>
    </xsd:extension>
</xsd:complexContent>
</xsd:complexType>
</xsd:element>

<xsd:element name="getJobPermissions">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmCommandBase">
                <xsd:sequence>
                    <xsd:element name="job_name"
minOccurs="1" type="xsd:string" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<xsd:element name="getJobPermissionsOutput">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmOutputBase">
                <xsd:sequence>
                    <xsd:element name="job_security"
minOccurs="0" maxOccurs="1" type="securityType" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<xsd:element name="getJobContextMenuPermission">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmCommandBase">
                <xsd:sequence>
                    <xsd:element name="job_name"
minOccurs="1" type="xsd:string" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

```

```

<xsd:element name="getJobContextMenuPermissionOutput">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="dmOutputBase">
        <xsd:sequence>
          <xsd:element
name="contextPermission" minOccurs="1" maxOccurs="1"
type="jobContextPermission" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<xsd:element name="getGlobalAccessPermissionByUserAndRoles">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="dmCommandBase">
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<xsd:element name="getGlobalAccessPermissionByUserAndRolesOutput">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="dmOutputBase">
        <xsd:sequence>
          <xsd:element
name="globalAccessPermission" minOccurs="1" maxOccurs="1"
type="accessPermissionType" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<xsd:element name="getGlobalModificationPermissionByUserAndRoles">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="dmCommandBase">
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

```

```

        </xsd:complexType>
    </xsd:element>

    <xsd:element name="getGlobalModificationPermissionByUserAndRolesOutput">
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="dmOutputBase">
                    <xsd:sequence>
                        <xsd:element
name="globalModificationPermission" minOccurs="1" maxOccurs="1"
type="modificationPermissionType" />
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>

    <xsd:simpleType name="permissionType">
        <!-- Restricting the values to a set of value using 'enumeration'
-->
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="read" />
            <xsd:enumeration value="execute" />
            <xsd:enumeration value="write" />
            <xsd:enumeration value="owner" />
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:element name="getJobsWithPermission">
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="dmCommandBase">
                    <xsd:sequence>
                        <xsd:element name="permission"
minOccurs="1" type="permissionType" />
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="getJobsWithPermissionOutput">
        <xsd:complexType>
            <xsd:complexContent>

```

```

        <xsd:extension base="dmOutputBase">
            <xsd:sequence>
                <xsd:element name="job_name"
minOccurs="0" maxOccurs="unbounded" type="xsd:string" />
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
</xsd:element>

<!-- Update permission type -->
<xsd:complexType name="updatePermissionType">
    <xsd:sequence>
        <xsd:element name="job_name" minOccurs="1" maxOccurs="1"
type="xsd:string" />
        <xsd:element name="job_security" minOccurs="1"
maxOccurs="1" type="securityType" />
    </xsd:sequence>
</xsd:complexType>

<xsd:element name="updateJobPermissions">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmCommandBase">
                <xsd:sequence>
                    <xsd:element
name="job_permission" minOccurs="1" maxOccurs="unbounded"
type="updatePermissionType" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<xsd:element name="updateJobPermissionsOutput">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmOutputBase">
                <xsd:sequence>
                    <xsd:element name="status"
maxOccurs="1" type="xsd:int" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

```

```

        </xsd:complexType>
    </xsd:element>

    <xsd:element name="removeRole">
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="dmCommandBase">
                    <xsd:sequence>
                        <xsd:element name="role"
maxOccurs="1" type="xsd:string" />
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="removeUser">
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="dmCommandBase">
                    <xsd:sequence>
                        <xsd:element name="username"
maxOccurs="1" type="xsd:string" />
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>

    <!-- Security type, view_permission is the same as read_permission,
provided for backward compatiability -->
    <xsd:complexType name="securityType">
        <xsd:sequence>
            <xsd:element name="owner_name" minOccurs="0"
maxOccurs="1" type="xsd:string" />
            <xsd:element name="read_permission" minOccurs="0"
maxOccurs="1" type="userSecurityType" />
            <xsd:element name="write_permission" minOccurs="0"
maxOccurs="1" type="userSecurityType" />
            <xsd:element name="execute_permission" minOccurs="0"
maxOccurs="1" type="userSecurityType" />
        </xsd:sequence>
    </xsd:complexType>

```

```

    <!-- User security type -->
    <xsd:complexType name="userSecurityType">
        <xsd:sequence>
            <xsd:element name="username" minOccurs="0"
maxOccurs="unbounded" type="xsd:string" />
            <xsd:element name="role" minOccurs="0"
maxOccurs="unbounded" type="xsd:string" />
        </xsd:sequence>
    </xsd:complexType>

    <!-- Define data types -->
    <!-- propertyType -->
    <xsd:complexType name="propertyType">
        <xsd:sequence>
            <xsd:element name="key" type="xsd:string" />
            <xsd:element name="value" type="valueType"
maxOccurs="unbounded" minOccurs="1"/>
            <xsd:element name="systemPairs" type="systemPairType"
maxOccurs="1" minOccurs="0"/>
            <xsd:element name="groupPools" type="groupPoolsType"
maxOccurs="1" minOccurs="0"/>
            <xsd:element name="targetUserPool"
type="targetUserPoolType" maxOccurs="1" minOccurs="0"/>
            <xsd:element name="systemList" type="neverTargetListType"
maxOccurs="1" minOccurs="0"/>
            <xsd:element name="defaultDatabases"
type="defaultDatabasesType" maxOccurs="1" minOccurs="0"/>
            <xsd:element name="unit" type="xsd:string" maxOccurs="1"
minOccurs="0"/>
            <xsd:element name="maps" type="mapsType" maxOccurs="1"
minOccurs="0"/>
            <xsd:element name="description" minOccurs="0"
type="xsd:string" />
        </xsd:sequence>
    </xsd:complexType>

    <!-- Unite Type -->
    <xsd:simpleType name="tmsmFrequencyBytesUnitType">
        <!-- Restricting the values to a set of value using 'enumeration'
-->
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="BYTES" />
            <xsd:enumeration value="MB" />

```



```

        <xsd:enumeration value="GB" />
    </xsd:restriction>
</xsd:simpleType>

<!-- mapsType -->
<xsd:complexType name="mapsType">
    <xsd:sequence>
        <xsd:element name="systemLevelMap" type="mapType" maxOccurs="unbounded"
minOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="mapType">
    <xsd:sequence>
        <xsd:element name="system" type="xsd:string" maxOccurs="1"
minOccurs="1"/>
        <xsd:element name="map" type="xsd:string" maxOccurs="1"
minOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>

<!-- defaultDatabasesType -->
<xsd:complexType name="defaultDatabasesType">
    <xsd:sequence>
        <xsd:element name="systemLevelDatabase"
type="systemLevelDatabaseType" maxOccurs="unbounded" minOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>

<!-- systemLevelDatabaseType -->
<xsd:complexType name="systemLevelDatabaseType">
    <xsd:sequence>
        <xsd:element name="system" type="xsd:string" maxOccurs="1"
minOccurs="1"/>
        <xsd:element name="source_staging_database"
type="xsd:string" maxOccurs="1" minOccurs="0"/>
        <!-- new tag to replace staging_database for target
staging database -->
        <xsd:element name="target_staging_database"
type="xsd:string" maxOccurs="1" minOccurs="0"/>
        <xsd:element name="staging_database" type="xsd:string"
maxOccurs="1" minOccurs="0"/>
        <xsd:element name="staging_database_for_table"

```

```

type="xsd:string" maxOccurs="1" minOccurs="0"/>
        <xsd:element name="target_database" type="xsd:string"
maxOccurs="1" minOccurs="0"/>

        </xsd:sequence>
</xsd:complexType>

<!-- neverTargetListType -->
<xsd:complexType name="neverTargetListType">
    <xsd:sequence>
        <xsd:element name="targetSystem" type="xsd:string"
maxOccurs="unbounded" minOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>

<!-- systemPairType -->
<xsd:complexType name="systemPairType">
    <xsd:sequence>
        <xsd:element name="forceDirectionPair"
type="forceDirectionPairType" maxOccurs="unbounded" minOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>

<!-- userGroupPoolType -->
<xsd:complexType name="groupPoolsType">
    <xsd:sequence>
        <xsd:element name="groupPool" type="userGroupType"
maxOccurs="unbounded" minOccurs="1"/>
    </xsd:sequence>

</xsd:complexType>

<!-- userGroupType -->
<xsd:complexType name="userGroupType">

    <xsd:sequence>
        <xsd:element name="name" type="xsd:string" maxOccurs="1"
minOccurs="1"/>
        <xsd:element name="system" type="systemType" maxOccurs="unbounded"
minOccurs="1"/>
    </xsd:sequence>

</xsd:complexType>

```

```

<!-- targetUserPoolType -->
<xsd:complexType name="targetUserPoolType">
  <xsd:sequence>
    <xsd:element name="system" type="systemType" maxOccurs="unbounded"
minOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<!-- systemType -->
<xsd:complexType name="systemType">
  <xsd:sequence>
    <xsd:element name="user" type="userType" maxOccurs="unbounded"
minOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string" use="required"/>
</xsd:complexType>

<!-- forceDirectionPairType -->
<xsd:complexType name="forceDirectionPairType">
  <xsd:sequence>
    <xsd:element name="sourceSystem" type="xsd:string" maxOccurs="1"
minOccurs="1"/>
    <xsd:element name="targetSystem" type="xsd:string"
maxOccurs="1" minOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="allowStatsBack" type="xsd:boolean"
use="required"/>
</xsd:complexType>

<!-- userType -->
<xsd:complexType name="userType">
  <xsd:sequence>
    <xsd:element name="name" type="xsd:string" maxOccurs="1"
minOccurs="1"/>
    <xsd:element name="password" type="xsd:string"
maxOccurs="1" minOccurs="0"/>
    <xsd:element name="encrypted_password" type="xsd:string"
maxOccurs="1" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<!-- valueType -->
<xsd:complexType name="valueType">
  <xsd:simpleContent>

```

```

        <xsd:extension base="simpleValueType">
            <xsd:attribute name="system" type="xsd:string"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>

<!-- simpleValueType -->
    <xsd:simpleType name="simpleValueType">
        <xsd:restriction base="xsd:string">
        </xsd:restriction>
    </xsd:simpleType>

<!-- startJobStatusType -->
    <xsd:simpleType name="startJobStatusType">
        <!-- Restricting the values to a set of value using 'enumeration'
-->
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="STARTED" />
            <xsd:enumeration value="QUEUED" />
            <xsd:enumeration value="BLOCKED" />
            <xsd:enumeration value="REJECTED" />
            <xsd:enumeration value="USER_CANCELLED" />
        </xsd:restriction>
    </xsd:simpleType>

<!-- stopJobStatusType -->
    <xsd:simpleType name="stopJobStatusType">
        <!-- Restricting the values to a set of value using 'enumeration'
-->
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="dropped_from_queue" />
            <xsd:enumeration value="stop_requested" />
            <xsd:enumeration value="not_running_nor_queued" />
            <xsd:enumeration value="not_running_but_new_instance" />
            <xsd:enumeration value="not_running_stopped_latest" />
            <xsd:enumeration value="blocked_job_stopped" />
        </xsd:restriction>
    </xsd:simpleType>

<!-- statusType -->
    <xsd:simpleType name="statusType">
        <!-- Restricting the values to a set of value using 'enumeration'
-->
        <xsd:restriction base="xsd:string">

```

```

        <xsd:enumeration value="INITIALIZING" />
        <xsd:enumeration value="RUNNING" />
        <xsd:enumeration value="BLOCKED" />
        <xsd:enumeration value="COMPLETED_SUCCESSFULLY" />
        <xsd:enumeration value="FAILED" />
        <xsd:enumeration value="RESTARTING" />
        <xsd:enumeration value="QUEUED" />
        <xsd:enumeration value="USER_CANCELLED" />
        <xsd:enumeration value="NEW" />
        <xsd:enumeration value="ARC_SOCKET_FAILURE" />
        <xsd:enumeration value="COMPLETED_WITH_ERRORS" />
        <xsd:enumeration value="COMPLETED_WITH_WARNINGS" />
    </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="stepEnumType">
    <!-- Restricting the values to a set of value using 'enumeration'
-->
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="NONE" />
        <xsd:enumeration value="CLEAN_UP" />
        <xsd:enumeration value="BUILD_DATABASE" />
        <xsd:enumeration value="MOVE_DATABASE_DEFINITION" />
        <xsd:enumeration value="MOVE_DEFINITION_BEFORE_LOAD" />
        <xsd:enumeration value="MOVE_DEF_AND_SRC_STG_DATA" />
        <xsd:enumeration value="MOVE_DATABASE_DATA" />
        <xsd:enumeration value="MOVE_TABLE_DATA" />
        <xsd:enumeration value="MOVE_VIEW_DATA" />
        <xsd:enumeration value="MOVE_JOURNAL_DATA" />
        <xsd:enumeration value="RESOLVE_TABLE_AFTER_LOAD" />
        <xsd:enumeration value="COPY_STATISTICS" />
        <xsd:enumeration value="HARD_DELETE" />
        <xsd:enumeration value="COMPARE_DDL"/>
        <xsd:enumeration value="VERIFY_JOB_OBJECTS"/>
        <xsd:enumeration value="ROW_COUNT_VALIDATION"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="objectMovePhaseType">
    <!-- Restricting the values to a set of value using 'enumeration'
-->
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="NOT_STARTED" />
        <xsd:enumeration value="PRE_DATA_MOVE" />

```

```

        <xsd:enumeration value="MOVING_DATA" />
        <xsd:enumeration value="MOVING_STAGING_TO_TARGET" />
        <xsd:enumeration value="POST_DATA_MOVE" />
        <xsd:enumeration value="VALIDATING" />
        <xsd:enumeration value="COMPLETED_SUCCESSFULLY" />
        <xsd:enumeration value="CLEAN_UP" />
        <xsd:enumeration value="BUILD_DATABASE" />
        <xsd:enumeration value="COMPARE_DDL"></xsd:enumeration>
        <xsd:enumeration value="ROW_COUNT_VALIDATION"></
xsd:enumeration>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="jobTaskPhaseType">
        <!-- Restricting the values to a set of value using 'enumeration'
-->
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="INITIATE" />
            <xsd:enumeration value="ACQUISITION" />
            <xsd:enumeration value="APPLY_ROWS" />
            <xsd:enumeration value="BUILD" />
            <xsd:enumeration value="TERMINATE" />
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="rowCountStatusEnumType">
        <!-- Restricting the values to a set of value using 'enumeration'
-->
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="IN_SYNC" />
            <xsd:enumeration value="OUT_OF_SYNC" />
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="objectType">
        <!-- Restricting the values to a set of value using 'enumeration'
-->
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="root" />
            <xsd:enumeration value="table" />
            <xsd:enumeration value="journal" />
            <xsd:enumeration value="stored_procedure" />
            <xsd:enumeration value="udf" />
            <xsd:enumeration value="macro" />

```

```

        <xsd:enumeration value="view" />
        <xsd:enumeration value="trigger" />
        <xsd:enumeration value="hash_index" />
        <xsd:enumeration value="join_index" />
        <xsd:enumeration value="queue_table" />
        <xsd:enumeration value="replication_group" />
        <xsd:enumeration value="database" />
        <xsd:enumeration value="no_pi_table" />
        <xsd:enumeration value="aggregate_udf" />
        <xsd:enumeration value="aggregate_ordered_function" />
        <xsd:enumeration value="jar" />
        <xsd:enumeration value="external_stored_procedure" />
        <xsd:enumeration value="instance_constructor_method" />
        <xsd:enumeration value="table_function" />
        <xsd:enumeration value="ordered_analytical_function" />
        <xsd:enumeration value="udt" />
        <xsd:enumeration value="authorization" />
        <xsd:enumeration value="glop" />
        <xsd:enumeration value="foreign_server" />
        <xsd:enumeration value="function_alias" />
        <xsd:enumeration value="aster_mr" />
        <xsd:enumeration value="unknown" />
        <xsd:enumeration value="blank" />
    </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="objectStatusEnumType">
    <!-- Restricting the values to a set of value using 'enumeration'
-->
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="NONE" />
        <xsd:enumeration value="ARCHIVE" />
        <xsd:enumeration value="COPY" />
        <xsd:enumeration value="EXPORT" />
        <xsd:enumeration value="LOAD" />
        <xsd:enumeration value="UPDATE" />
        <xsd:enumeration value="STREAM" />
        <xsd:enumeration value="COMPLETE" />
        <xsd:enumeration value="FAILED" />
        <xsd:enumeration value="INSERT" />
        <xsd:enumeration value="COMPLETED_WITH_ERRORS" />
        <xsd:enumeration value="SELECT" />
    </xsd:restriction>
</xsd:simpleType>

```

```

<xsd:simpleType name="TPTAPIOperatorType">
  <!-- Restricting the values to a set of value using 'enumeration'
-->
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="NONE" />
    <xsd:enumeration value="TPTAPI_LOAD" />
    <xsd:enumeration value="TPTAPI_EXPORT" />
    <xsd:enumeration value="TPTAPI_UPDATE" />
    <xsd:enumeration value="TPTAPI_STREAM" />
  </xsd:restriction>
</xsd:simpleType>

<!-- stepsType -->
<xsd:complexType name="jobStepType">
  <xsd:sequence>
    <xsd:element name="start_time" type="xsd:dateTime"
minOccurs="0" />
    <xsd:element name="end_time" type="xsd:dateTime"
minOccurs="0" />
    <xsd:element name="order" type="xsd:int" />
    <xsd:element name="job_step_ID" type="xsd:long" />
    <xsd:element name="step" type="stepEnumType" />
    <xsd:element name="status" type="statusType" />
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="taskStatusType">
  <xsd:sequence>
    <xsd:element name="jobPlanId" type="xsd:long" />
    <xsd:element name="jobTaskId" type="xsd:long" />
    <xsd:element name="timeStamp" type="xsd:dateTime" />
    <xsd:element name="parentName" type="xsd:string" />
    <xsd:element name="objectName" type="xsd:string" />
    <xsd:element name="agentName" type="xsd:string"
nillable="true" />
    <xsd:element name="sequence" type="xsd:long" />
    <xsd:element name="bytesExported" type="xsd:long" />
    <xsd:element name="rowsExported" type="xsd:long" />
    <xsd:element name="bytesInserted" type="xsd:long" />
    <xsd:element name="rowsInserted" type="xsd:long" />
    <xsd:element name="bytesUpdated" type="xsd:long" />
    <xsd:element name="rowsUpdated" type="xsd:long" />
    <xsd:element name="bytesDeleted" type="xsd:long" />
  </xsd:sequence>
</xsd:complexType>

```



```

<xsd:element name="rowsDeleted" type="xsd:long" />
<xsd:element name="activityCount" type="xsd:long" />
<xsd:element name="totalRowsProcessed" type="xsd:long" />
<xsd:element name="totalBytesProcessed"
type="xsd:long" />
<xsd:element name="sourceTableSize" type="xsd:long" />
<xsd:element name="sourceTableRowCount"
type="xsd:long" />
<xsd:element name="targetTableSize" type="xsd:long" />
<xsd:element name="targetTableRowCount"
type="xsd:long" />
<xsd:element name="errorRowsCount" type="xsd:long" />
<xsd:element name="status" type="objectMovePhaseType" />
<xsd:element name="type" type="objectStatusEnumType" />
<xsd:element name="object" type="objectType" />
<xsd:element name="taskStatus" type="statusType"
minOccurs="0" />
<xsd:element name="errorCode" type="xsd:int" />
<xsd:element name="errorMsg" type="xsd:string"
minOccurs="0" />
<xsd:element name="sequenceNumber" type="xsd:int" />
<xsd:element name="status_source" type="xsd:string"
minOccurs="0" />
<xsd:element name="taskType" type="xsd:string"
minOccurs="0" />
<xsd:element name="jobTaskPhase" type="jobTaskPhaseType"
maxOccurs="1" minOccurs="0"/>
<xsd:element name="tptid" type="xsd:string"
minOccurs="0"/>
<xsd:element name="sourceSessions" type="xsd:int"
default="-1" minOccurs="0"/>
<xsd:element name="targetSessions" type="xsd:int"
default="-1" minOccurs="0"/>
<xsd:element name="dataStreams" type="xsd:int"
default="-1" minOccurs="0"/>
<xsd:element name="utility" type="xsd:string"
minOccurs="0"/>
<xsd:element name="actualSourceSessions" type="xsd:int"
default="-1" minOccurs="0"/>
<xsd:element name="actualTargetSessions" type="xsd:int"
default="-1" minOccurs="0"/>
<xsd:element name="startTime" type="xsd:dateTime"
minOccurs="0"/>
<xsd:element name="stopTime" type="xsd:dateTime"

```

```

minOccurs="0"/>
        <xsd:element name="runtimeInfo" type="xsd:string"
minOccurs="0"/>
        <xsd:element name="postLoadSqlList" minOccurs="0">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="postLoadSql"
type="PostLoadSQLType"
                                minOccurs="0"
maxOccurs="unbounded" />
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="syncBarrierMessageType">
    <xsd:sequence>
        <xsd:element name="jobTaskId" type="xsd:long" />
        <xsd:element name="instanceId" type="xsd:int" />
        <xsd:element name="operatorType" type="TPTAPIOperatorType" />
        <xsd:element name="endMethod" type="xsd:boolean" />
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="syncTelinfoMessageType">
    <xsd:sequence>
        <xsd:element name="jobTaskId" type="xsd:long" />
        <xsd:element name="instanceId" type="xsd:int" />
        <xsd:element name="operatorType" type="TPTAPIOperatorType" />
        <xsd:element name="telinfoLen" type="xsd:long" />
        <xsd:element name="telinfo" type="xsd:byte"
maxOccurs="unbounded" minOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="syncSchemaMessageType">
    <xsd:sequence>
        <xsd:element name="jobTaskId" type="xsd:long" />
        <xsd:element name="instanceId" type="xsd:int" />
        <xsd:element name="operatorType" type="TPTAPIOperatorType" />
        <xsd:element name="schemaLen" type="xsd:long" />
        <xsd:element name="schemaNumCols" type="xsd:long" />
        <xsd:element name="schemaCols" type="xsd:byte"

```

```

maxOccurs="unbounded" minOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="taskRowCountStatusType">
    <xsd:sequence>
        <xsd:element name="jobPlanID" type="xsd:long" />
        <xsd:element name="jobStepID" type="xsd:long" />
        <xsd:element name="timeStamp" type="xsd:dateTime" />
        <xsd:element name="parentName" type="xsd:string" />
        <xsd:element name="objectName" type="xsd:string" />
        <xsd:element name="targetParentName" type="xsd:string" />
        <xsd:element name="targetObjectName" type="xsd:string" />
        <xsd:element name="sequence" type="xsd:long" />
        <xsd:element name="sourceRowCount" type="xsd:long" />
        <xsd:element name="targetRowCount" type="xsd:long" />
        <xsd:element name="status"
type="rowCountStatusEnumType" />
        <xsd:element name="validationtype"
type="RowCountValidationType" />
        <xsd:element name="errorString" type="xsd:string" />
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="streamProcessInfoType">
    <xsd:sequence>
        <xsd:element name="utility" type="xsd:string" />
        <xsd:element name="jobTaskID" type="xsd:long" />
        <xsd:element name="agentName" type="xsd:string" />
        <xsd:element name="nodeName" type="xsd:string"
minOccurs="0"/>
        <xsd:element name="operator" type="xsd:string" />
        <xsd:element name="streamID" type="xsd:int" />
        <xsd:element name="bytesProcessed" type="xsd:long" />
        <xsd:element name="processSpeed" type="xsd:long" />
        <xsd:element name="timestamp" type="xsd:dateTime" />
        <xsd:element name="objectName" type="xsd:string" />
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="jobExecutionLogLineInfoType">
    <xsd:sequence>
        <xsd:element name="jobExecutionId" type="xsd:long" />
        <xsd:element name="jobDefId" type="xsd:long" />

```

```

        <xsd:element name="eventType" type="xsd:string" />
        <xsd:element name="startTime" type="xsd:dateTime" />
        <xsd:element name="stopTime" type="xsd:dateTime" />
        <xsd:element name="logTime" type="xsd:dateTime" />
        <xsd:element name="sequence" type="xsd:long" />
        <xsd:element name="logText" type="xsd:string" />
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="LockedObjectStatusType">
    <xsd:sequence>
        <xsd:element name="parentName" type="xsd:string" />
        <xsd:element name="objectName" type="xsd:string" />
        <xsd:element name="lockOnSource" type="xsd:boolean" />
        <xsd:element name="objectType" type="xsd:string" />
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="blockedJobStatusType">
    <xsd:sequence>
        <xsd:element name="jobDefinitionID" type="xsd:long" />
        <xsd:element name="jobExecutionID" type="xsd:long" />
        <xsd:element name="lockedObjectsStatusType"
type="LockedObjectStatusType"
            maxOccurs="unbounded" minOccurs="0" />
        <xsd:element name="lastCheckTimeStamp"
type="xsd:dateTime" />
        <xsd:element name="blockStartTimeStamp"
type="xsd:dateTime" />
        <xsd:element name="lockedSystem" type="xsd:string" />
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="logLineType">
    <xsd:sequence>
        <xsd:element name="log" type="xsd:string" />
        <xsd:element name="streamIndex" type="xsd:long" />
    </xsd:sequence>
</xsd:complexType>

<!-- jobExecutionType -->
<xsd:complexType name="jobExecutionType">
    <xsd:sequence>
        <xsd:element name="job_name" type="xsd:string" />

```

```

minOccurs="0"/>
<xsd:element name="job_execution_name" type="xsd:string"
minOccurs="0" />
<xsd:element name="job_size" type="xsd:long"
minOccurs="0" />
<xsd:element name="job_instance_ID" type="xsd:long" />
<xsd:element name="job_execution_ID" type="xsd:long" />
<xsd:element name="job_plan_ID" type="xsd:long" />
<xsd:element name="current_step" type="xsd:long" />
<xsd:element name="restart_plan_ID" type="xsd:long" />
<xsd:element name="submit_time" type="xsd:dateTime"
minOccurs="0" />
<xsd:element name="start_time" type="xsd:dateTime"
minOccurs="0" />
<xsd:element name="end_time" type="xsd:dateTime"
minOccurs="0" />
<xsd:element name="status" type="statusType" />
<xsd:element name="user_name" type="xsd:string"
minOccurs="0"/>
<xsd:element name="job_priority"
type="jobPriorityType" />
<xsd:element name="job_step" type="jobStepType"
maxOccurs="unbounded" minOccurs="0" />
<xsd:element name="task_status" type="taskStatusType"
maxOccurs="unbounded" minOccurs="0" />
<xsd:element name="row_count_status"
type="taskRowCountStatusType"
maxOccurs="unbounded" minOccurs="0" />
<xsd:element name="blocked_job_status"
type="blockedJobStatusType" nillable="true"/>
<xsd:element name="log_line" type="logLineType"
maxOccurs="unbounded" minOccurs="0" />
<xsd:element name="stream_process_info"
type="streamProcessInfoType"
maxOccurs="unbounded" minOccurs="0" />
<xsd:element name="job_execution_log_line_info"
type="jobExecutionLogLineInfoType"
maxOccurs="unbounded" minOccurs="0" />
</xsd:sequence>
</xsd:complexType>

<!-- listJobsRecordType -->
<xsd:complexType name="listJobsRecordType">
<xsd:sequence>
<xsd:element name="job_name" type="xsd:string" />

```

```

        <xsd:element name="job_execution_name" type="xsd:string"
minOccurs="0"/>
        <xsd:element name="submit_time" type="xsd:dateTime"
minOccurs="0"/>
        <xsd:element name="start_time" type="xsd:dateTime"
minOccurs="0"/>
        <xsd:element name="end_time" type="xsd:dateTime"
minOccurs="0"/>
        <xsd:element name="status" type="statusType" />
        <xsd:element name="job_priority"
type="jobPriorityType" />
    </xsd:sequence>
</xsd:complexType>

<!-- agentAnnouncementType -->
<xsd:complexType name="agentInfoType">
    <xsd:sequence>
        <xsd:element name="agent_id" type="xsd:string"/>
        <xsd:element name="agent_status" type="agentStatusType"/>
        <xsd:element name="agent_hostname" type="xsd:string"/>
        <xsd:element name="agent_username" type="xsd:string"/>
        <xsd:element name="agent_public_key" type="xsd:string"/>
        <xsd:element name="agent_max_concurrent_tasks"
type="xsd:int"/>
        <xsd:element name="agent_current_task_count"
type="xsd:int"/>
        <xsd:element name="agent_monitor_property_file_change"
type="xsd:boolean" default="true"/>
        <xsd:element name="agent_uuid" type="xsd:string" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>

<!-- sessionsAndStreamsType -->
<xsd:complexType name="sessionsAndStreamsType">
    <xsd:simpleContent>
        <xsd:extension base="xsd:int">
            <xsd:attribute name="type" type="xsd:string"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>

<!-- Agent and Daemon -->

```

```

<!-- dmAgentAnnouncement -->
<xsd:element name="dmAgentAnnouncement">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="dmCommandBase">
        <xsd:sequence>
          <xsd:element name="agent_info"
type="agentInfoType"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<!-- dmAgent Unexpected Error -->
<xsd:element name="dmAgentUnexpectedError">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="dmAgentCommandBase">
        <xsd:sequence>
          <xsd:element name="errorMessage"
type="xsd:string"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<xsd:simpleType name="agentStatusType">
  <!-- Restricting the values to a set of value using 'enumeration'
-->
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="started" />
    <xsd:enumeration value="stopped" />
  </xsd:restriction>
</xsd:simpleType>

<!-- DMPingAgent -->
<xsd:element name="dmPingAgent">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="dmAgentCommandBase">
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

```

```

        </xsd:complexType>
    </xsd:element>

    <!-- DMPingAgent -->
    <xsd:element name="dmUpdateAgentProperty">
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="dmAgentCommandBase">
                    <xsd:sequence>
                        <xsd:element
name="enable_monitor_agent_property_change" minOccurs="0" type="xsd:boolean"
default="false" />
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>

    <!-- SQLTask -->
    <xsd:element name="dmSQLTask">
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="dmAgentCommandBase">
                    <xsd:sequence>
                        <xsd:element
name="target_db_type" minOccurs="0" type="xsd:string" />
                        <xsd:element
name="target_db_name" minOccurs="0" type="xsd:string" />
                        <xsd:element
name="target_db_port" minOccurs="0" type="xsd:string" />
                        <xsd:element
name="target_user_name" minOccurs="0" type="xsd:string" />
                        <xsd:element
name="target_system_name" minOccurs="0" type="xsd:string" />
                        <xsd:element
name="target_logon_mech" minOccurs="0" type="xsd:string" />
                        <xsd:element
name="target_logon_mech_data" minOccurs="0" type="xsd:string" />
                        <xsd:element
name="target_account_id" type="xsd:string" minOccurs="0" />
                        <xsd:element
name="source_user_name" minOccurs="0" type="xsd:string" />
                        <xsd:element
name="source_system_name" minOccurs="0" type="xsd:string" />
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>

```



```

                                <xsd:element
name="source_logon_mech" minOccurs="0" type="xsd:string" />
                                <xsd:element
name="source_logon_mech_data" minOccurs="0" type="xsd:string" />
                                <xsd:element
name="source_account_id" minOccurs="0" type="xsd:string" />
                                <xsd:element
name="sql_statement_list" type="sqlStatementType" minOccurs="0"
maxOccurs="unbounded" />
                                <xsd:element
name="source_sql_statement_list" type="sqlStatementType" minOccurs="0"
maxOccurs="unbounded" />
                                <xsd:element name="object_list"
type="tableType" minOccurs="0" maxOccurs="unbounded" />
                                <xsd:element
name="stats_task_list" type="statsTaskType" minOccurs="0"
maxOccurs="unbounded" />
                                <xsd:element name="job_task"
type="jobTaskType" />
                                <xsd:element name="move_phase"
type="objectMovePhaseType" />
                                <xsd:element name="clean_up"
type="xsd:boolean" default="false" />
                                <xsd:element name="copy_stats"
type="xsd:boolean" default="false" />
                                <xsd:element
name="database_client_encryption" type="xsd:boolean" default="false" />
                                <xsd:element
name="data_encrypted_by_agent" nillable="true" maxOccurs="unbounded"
type="dmAgentEncryptedBase" />
                                <xsd:element
name="source_session_charset" minOccurs="0" type="xsd:string"></xsd:element>
                                <xsd:element
name="target_session_charset" minOccurs="0" type="xsd:string"></xsd:element>
                                <xsd:element
name="deadlock_retry_enabled" minOccurs="0" type="xsd:boolean" default="false"></
xsd:element>
                                <xsd:element
name="deadlock_retry_interval" minOccurs="0" type="xsd:long" default="0"></
xsd:element>
                                <xsd:element
name="deadlock_retry_max_attempts" minOccurs="0" type="xsd:int" default="0"></
xsd:element>
                                <xsd:element

```

```

name="source_jdbc_connection_string" type="xsd:string" minOccurs="0"/>
    <xsd:element
name="target_jdbc_connection_string" type="xsd:string" minOccurs="0"/>

    </xsd:sequence>
  </xsd:extension>
</xsd:complexContent>
</xsd:complexType>
</xsd:element>

<!-- AsterMRTask -->
  <xsd:element name="dmAsterMRTask">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="dmAgentCommandBase">
          <xsd:sequence>

            <xsd:element
name="source_db_type" minOccurs="0"

              type="xsd:string" />
            <xsd:element
name="source_db_name" minOccurs="0"

              type="xsd:string" />
            <xsd:element
name="source_db_port" minOccurs="0"

              type="xsd:string" />

            <xsd:element
name="source_user_name" minOccurs="0"

              type="xsd:string" />
            <xsd:element
name="source_system_name" minOccurs="0"

              type="xsd:string" />
            <xsd:element
name="source_logon_mech" minOccurs="0"

              type="xsd:string" />
            <xsd:element
name="source_logon_mech_data" minOccurs="0"

              type="xsd:string" />
            <xsd:element
name="source_account_id" type="xsd:string" minOccurs="0" />
            <xsd:element
name="source_session_charset" minOccurs="0" type="xsd:string"></xsd:element>

```

```

name="source_schema_name" minOccurs="0" type="xsd:string" />
name="source_parent"
minOccurs="0" type="xsd:string" />
name="source_table"

name="target_db_type" minOccurs="0"
type="xsd:string" />
name="target_db_name" minOccurs="0"
type="xsd:string" />
name="target_db_port" minOccurs="0"
type="xsd:string" />

name="target_user_name" minOccurs="0"
type="xsd:string" />
name="target_system_name" minOccurs="0"
type="xsd:string" />
name="target_logon_mech" minOccurs="0"
type="xsd:string" />
name="target_logon_mech_data" minOccurs="0"
type="xsd:string" />
name="target_account_id" type="xsd:string" minOccurs="0" />
name="target_session_charset" minOccurs="0" type="xsd:string"></xsd:element>

name="target_schema_name" minOccurs="0" type="xsd:string" />
name="target_parent"
minOccurs="0" type="xsd:string" />
name="target_table"


```

```

name="td_error_table_name" minOccurs="0" type="xsd:string" />
                                <xsd:element
name="td_ucv_table_name" minOccurs="0" type="xsd:string" />

                                <xsd:element
name="sql_mr_statement" type="xsd:string" />

                                <xsd:element
name="aster_progress_query" minOccurs="0"
                                type="xsd:string" />

                                <xsd:element
name="teradata_progress_size" minOccurs="0"
                                type="xsd:string" />

                                <xsd:element
name="teradata_progress_rowcount_size" minOccurs="0"
                                type="xsd:string" />
                                <xsd:element
name="dim_table_name" minOccurs="0"
                                type="xsd:string" />
                                <xsd:element
name="vworkers_count" type="xsd:int" minOccurs="0"
                                default="0" />

                                <xsd:element name="amps_count"
type="xsd:int" minOccurs="0"
                                default="0" />

                                <!--
                                <xsd:element
name="sql_statement_list" type="sqlStatementType"
                                minOccurs="0"
maxOccurs="unbounded" />

                                <xsd:element name="object_list"
type="tableType"
                                minOccurs="0"
maxOccurs="unbounded" />

                                -->
                                <xsd:element name="object_list"
type="tableType"
                                minOccurs="0"
maxOccurs="unbounded" />

```

```

type="jobTaskType" />
type="objectMovePhaseType" />
type="xsd:boolean" default="false" />
name="database_client_encryption" type="xsd:boolean" default="false" />
name="data_encrypted_by_agent" nillable="true"
maxOccurs="unbounded"
type="dmAgentEncryptedBase" />

<xsd:element
name="deadlock_retry_enabled" minOccurs="0" type="xsd:boolean" default="false"></
xsd:element>
<xsd:element
name="deadlock_retry_interval" minOccurs="0" type="xsd:long" default="0"></
xsd:element>
<xsd:element
name="deadlock_retry_max_attempts" minOccurs="0" type="xsd:int" default="0"></
xsd:element>
</xsd:sequence>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>
</xsd:element>

<!-- TPTAPITask -->
<xsd:element name="dmTPTAPIPingTask">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="dmAgentCommandBase">
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<!-- TPTAPITask -->
<xsd:element name="dmTPTAPITask">
  <xsd:complexType>

```

```

        <xsd:complexContent>
            <xsd:extension base="dmAgentCommandBase">
                <xsd:sequence>
                    <xsd:element name="source_tdpid"
type="xsd:string" />
                    <xsd:element
name="source_operator_type" type="TPTAPIOperatorType"
                        default="NONE" />
                    <xsd:element name="status"
minOccurs="0" type="xsd:string" />
                    <xsd:element
name="source_username" type="xsd:string" />
                    <xsd:element
name="source_logon_mech" type="xsd:string" minOccurs="0" />
                    <xsd:element
name="source_logon_mech_data" type="xsd:string" minOccurs="0" />
                    <xsd:element
name="source_account_id" type="xsd:string" minOccurs="0" />
                    <xsd:element
name="source_logon_mechanism" type="xsd:string" minOccurs="0" />
                    <xsd:element
name="source_logon_mechanism_data" type="xsd:string" minOccurs="0" />
                    <xsd:element
name="source_select_statement" type="sqlStatementType" />
                    <xsd:element name="source_parent"
type="xsd:string" />
                    <xsd:element name="source_table"
type="xsd:string" />
                    <xsd:element
name="source_sessions_count" type="xsd:int"
                        default="1" />
                    <xsd:element
name="source_instance_count" type="xsd:int"
                        default="1" />
                    <xsd:element
name="source_export_without_spool" type="xsd:int" />
                    <xsd:element
name="target_operator_type" type="TPTAPIOperatorType"
                        default="NONE" />
                    <xsd:element name="target_tdpid"
type="xsd:string" />
                    <xsd:element
name="target_username" type="xsd:string" />
                    <xsd:element

```

```

name="target_account_id" type="xsd:string" minOccurs="0" />
                                <xsd:element
name="target_logon_mechanism" type="xsd:string" minOccurs="0" />
                                <xsd:element
name="target_logon_mechanism_data" type="xsd:string" minOccurs="0" />
                                <xsd:element name="target_parent"
type="xsd:string" />
                                <xsd:element name="target_table"
type="xsd:string" />
                                <xsd:element
name="target_log_table" type="xsd:string" />
                                <xsd:element
name="target_error_table1" type="xsd:string" />
                                <xsd:element
name="target_error_table2" type="xsd:string" />
                                <xsd:element
name="target_work_table" type="xsd:string" />
                                <xsd:element
name="target_dml_statement" type="sqlStatementType"
                                minOccurs="0"
maxOccurs="unbounded" />
                                <xsd:element
name="target_sessions_count" type="xsd:int"
                                default="1" />
                                <xsd:element
name="target_instance_count" type="xsd:int"
                                default="1" />
                                <xsd:element
name="performance_report_rate" type="xsd:int"
                                default="0" />
                                <xsd:element name="log_level"
minOccurs="0" type="xsd:int" />
                                <xsd:element
name="log_file_name_preface" type="xsd:string" />
                                <xsd:element name="log_dir"
minOccurs="0" type="xsd:string" />
                                <xsd:element
name="database_client_encryption" type="xsd:boolean" default="false" />
                                <xsd:element
name="session_charset" type="xsd:string" />
                                <xsd:element name="table_type"
type="objectType" />
                                <xsd:element name="job_task"
type="jobTaskType" />

```

```

name="data_encrypted_by_agent" nillable="true"
                                maxOccurs="unbounded"
type="dmAgentEncryptedBase" />
                                <xsd:element name="instance_id"
                                default="1" maxOccurs="1"
minOccurs="1"/>
                                <xsd:element
name="instance_id_list" type="xsd:int"
                                maxOccurs="unbounded"
minOccurs="0"/>
                                <xsd:element name="max_instances"
                                default="1" />
                                <xsd:element name="cleanUp_agent"
type="xsd:boolean" default="false" />
                                <xsd:element
name="clean_up_job_plan_id" type="xsd:long" default="0" />
                                <xsd:element
name="clean_up_job_task_id" type="xsd:long" default="0" />
                                <xsd:element name="move_phase"
type="objectMovePhaseType" />
                                <xsd:element
name="source_session_charset" minOccurs="0" type="xsd:string"></xsd:element>
                                <xsd:element
name="target_session_charset" minOccurs="0" type="xsd:string"></xsd:element>
                                <xsd:element
name="source_char_encoding" minOccurs="0" type="xsd:string"></xsd:element>
                                <xsd:element
name="target_char_encoding" minOccurs="0" type="xsd:string"></xsd:element>
                                <xsd:element
name="source_unicodepassthrough" minOccurs="0" type="xsd:string"></xsd:element>
                                <xsd:element
name="target_unicodepassthrough" minOccurs="0" type="xsd:string"></xsd:element>
                                <xsd:element name="cliNetrace"
type="xsd:int" minOccurs="0" />
                                <xsd:element
name="cliNetraceBufLen" type="xsd:int" minOccurs="0" />
                                <xsd:element name="tptDebug"
type="xsd:int" minOccurs="0" />
                                <xsd:element name="source_tpt_connection" type="xsd:string"
minOccurs="0" />
                                <xsd:element name="target_tpt_connection" type="xsd:string"

```



```

minOccurs="0" />
                                </xsd:sequence>
                                </xsd:extension>
                                </xsd:complexContent>
                                </xsd:complexType>
                                </xsd:element>

<!-- JDBCTask -->
<xsd:element name="dmJDBCTask">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmAgentCommandBase">
                <xsd:sequence>
                    <xsd:element name="source_tdpid"
                                type="xsd:string" />
                    <xsd:element
name="source_username"
                                type="xsd:string" />
                    <xsd:element
name="source_logon_mech"
                                type="xsd:string"
minOccurs="0" />
                    <xsd:element
name="source_logon_mech_data"
                                type="xsd:string"
minOccurs="0" />
                    <xsd:element
name="source_account_id"
                                type="xsd:string"
minOccurs="0" />
                    <xsd:element
name="source_select_statement"
                                type="xsd:string" />
                    <xsd:element name="source_parent"
                                type="xsd:string" />
                    <xsd:element name="source_table"
                                type="xsd:string" />
                    <xsd:element name="target_tdpid"
                                type="xsd:string" />
                    <xsd:element
name="target_username"
                                type="xsd:string" />
                    <xsd:element
name="target_logon_mech"

```

```

minOccurs="0" />

name="target_logon_mech_data"

minOccurs="0" />

name="target_account_id"

minOccurs="0" />

name="target_log_table"

minOccurs="0" />

name="target_error_table1"

minOccurs="0" />

name="target_error_table2"

minOccurs="0" />

name="target_work_table"

minOccurs="0" />

name="target_insert_statement"

name="is_fast_export_usable"

default="false" />

name="is_fast_load_usable"

default="false" />

name="use_encryption"

type="xsd:string"

<xsd:element

type="xsd:string"

<xsd:element

type="xsd:string"

<xsd:element name="target_parent"

type="xsd:string" />

<xsd:element name="target_table"

type="xsd:string" />

<xsd:element

type="xsd:string"

<xsd:element

type="xsd:string"

<xsd:element

type="xsd:string"

<xsd:element

type="xsd:string" />

<xsd:element

type="xsd:boolean"

<xsd:element

type="xsd:boolean"

<xsd:element

type="xsd:boolean"

```

```

default="false" />
                                <xsd:element
name="performance_report_rate"
                                type="xsd:int"
default="0" />
                                <xsd:element name="session_charset" type="xsd:string"></
xsd:element>
                                <xsd:element name="table_type"
                                type="objectType" />
                                <xsd:element name="job_task"
type="jobTaskType" />
                                <xsd:element
name="data_encrypted_by_agent" nillable="true"
                                maxOccurs="unbounded"
type="dmAgentEncryptedBase" />
                                <xsd:element
name="source_session_charset" minOccurs="0" type="xsd:string"></xsd:element>
                                <xsd:element
name="target_session_charset" minOccurs="0" type="xsd:string"></xsd:element>
                                <xsd:element
name="database_client_encryption" type="xsd:boolean" default="false" />
                                <xsd:element name="move_phase"
type="objectMovePhaseType" />
                                <xsd:element
name="source_jdbc_connection_string" type="xsd:string" minOccurs="0"/>
                                <xsd:element
name="target_jdbc_connection_string" type="xsd:string" minOccurs="0"/>
                                </xsd:sequence>
                                </xsd:extension>
                                </xsd:complexContent>
                                </xsd:complexType>
                                </xsd:element>

                                <!-- DSATask -->
                                <xsd:element name="dmDSATask">
                                    <xsd:complexType>
                                        <xsd:complexContent>
                                            <xsd:extension base="dmAgentCommandBase">
                                                <xsd:sequence>
                                                    <xsd:element name="log_level"
minOccurs="0" type="xsd:int" />
                                                    <xsd:element name="dsa_job_name"
type="xsd:string" />
                                                    <xsd:element name="source_user"

```

```

type="xsd:string" />
<xsd:element name="target_user"
type="xsd:string" />
<xsd:element
name="job_model_json" type="xsd:string" nillable="true"/>
<xsd:element name="url"
type="xsd:string" />
<xsd:element name="dmUrl" type="xsd:string" />
<xsd:element name="isDscColocateDM" type="xsd:boolean" />
<xsd:element
name="shared_pipe_target_group" type="xsd:string"
minOccurs="0" />
<xsd:element name="job_task"
type="jobTaskType" />
<xsd:element
name="data_encrypted_by_agent" nillable="true"
maxOccurs="unbounded"
type="dmAgentEncryptedBase" />
<xsd:element
name="database_client_encryption" type="xsd:boolean"
default="false" />
<xsd:element
name="performance_report_rate" type="xsd:int" default="0" />
<xsd:element name="move_phase"
type="objectMovePhaseType" />
<xsd:element name="server_name"
minOccurs="0" type="xsd:string" />
<xsd:element name="groupUserPool"
minOccurs="0" type="xsd:string" />
</xsd:sequence>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>
</xsd:element>

<xsd:element name="dmCS2Task">
<xsd:complexType>
<xsd:complexContent>
<xsd:extension base="dmAgentCommandBase">
<xsd:sequence>
<xsd:element name="log_level" minOccurs="0"
type="xsd:int" />
<xsd:element name="cs2_job_name" type="xsd:string" />
<xsd:element name="source_user" type="xsd:string" />

```

```
<xsd:element name="target_user" type="xsd:string" />
<xsd:element name="job_model_json" type="xsd:string"
nillable="true"/>
<xsd:element name="url" type="xsd:string" />
<xsd:element name="dmUrl" type="xsd:string" />
<xsd:element name="job_task" type="jobTaskType" />
<xsd:element name="data_encrypted_by_agent"
nillable="true"
maxOccurs="unbounded"
type="dmAgentEncryptedBase" />
<xsd:element name="database_client_encryption"
type="xsd:boolean"
default="false" />
<xsd:element name="performance_report_rate" type="xsd:int"
default="0" />
<xsd:element name="move_phase"
type="objectMovePhaseType" />
<xsd:element name="server_name" minOccurs="0"
type="xsd:string" />
<xsd:element name="groupUserPool" minOccurs="0"
type="xsd:string" />
</xsd:sequence>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>
</xsd:element>

<!-- T2TTask -->
<xsd:element name="dmT2TTask">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="dmAgentCommandBase">
        <xsd:sequence>
          <xsd:element name="t2t_statement"
type="sqlStatementType" />
          <xsd:element name="source_tdpid"
            type="xsd:string" />
          <xsd:element
            type="xsd:string" />
          <xsd:element
name="source_username"
            type="xsd:string" />
          <xsd:element
name="source_logon_mech"
            type="xsd:string" />

```

```

minOccurs="0" />

name="source_logon_mech_data"

minOccurs="0" />

name="source_account_id"

minOccurs="0" />

name="target_username"

name="target_logon_mech"

minOccurs="0" />

name="target_logon_mech_data"

minOccurs="0" />

name="target_account_id"

minOccurs="0" />

name="use_encryption"

default="false" />

name="performance_report_rate"

default="0" />

<xsd:element name="session_charset" type="xsd:string"></
xsd:element>

```

```

<xsd:element
    type="xsd:string"

<xsd:element
    type="xsd:string"

<xsd:element name="source_parent"
    type="xsd:string" />
<xsd:element name="source_table"
    type="xsd:string" />
<xsd:element name="target_tdpid"
    type="xsd:string" />
<xsd:element
    type="xsd:string" />
<xsd:element
    type="xsd:string"

<xsd:element
    type="xsd:string"

<xsd:element
    type="xsd:string"

<xsd:element name="target_parent"
    type="xsd:string" />
<xsd:element name="target_table"
    type="xsd:string" />
<xsd:element
    type="xsd:boolean"

<xsd:element
    type="xsd:int"

```

```

        <xsd:element name="table_type"
                    type="objectType" />
        <xsd:element name="job_task"
                    type="jobTaskType" />
        <xsd:element
name="data_encrypted_by_agent" nillable="true"
                    maxOccurs="unbounded"
                    type="dmAgentEncryptedBase" />
        <xsd:element
name="database_client_encryption" type="xsd:boolean" default="false" />
        <xsd:element name="move_phase"
                    type="objectMovePhaseType" />
    </xsd:sequence>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>
</xsd:element>

    <xsd:element name="dmSysCmdTask">
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="dmAgentCommandBase">
                    <xsd:sequence>
                        <xsd:element name="sys_cmd_list"
                                    type="sysCmdType"
                                    maxOccurs="unbounded" />
                        <xsd:element name="job_task"
                                    type="jobTaskType" />
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>

    <xsd:complexType name="keyValueStringPair">
        <xsd:sequence>
            <xsd:element name="key" type="xsd:string"
                        nillable="false" />
            <xsd:element name="value" type="xsd:string"
                        nillable="false" />
        </xsd:sequence>
    </xsd:complexType>

    <!--

```

```

        <xsd:element name="dmAgentTask"> <xsd:complexType>
        <xsd:complexContent> <xsd:extension base="dmCommandBase">
        <xsd:sequence> <xsd:element name="agent_ID" minOccurs="0"
minOccurs="0"
        type="xsd:string" /> <xsd:element name="job_plan_ID"
        type="xsd:long" /> <xsd:element name="job_step_ID" minOccurs="0"
        type="xsd:long" /> <xsd:element name="job_task_ID" minOccurs="0"
        type="xsd:long" /> </xsd:sequence> </xsd:extension>
        </xsd:complexContent> </xsd:complexType> </xsd:element>

-->

    <xsd:element name="dmJobTask">
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="dmOutputBase">
                    <xsd:sequence>
                        <xsd:element name="job_task"
type="jobTaskType" />
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>

    <xsd:complexType name="jobTaskType">
        <xsd:sequence>
            <xsd:element name="jobPlanID" type="xsd:long"
default="0" />
            <xsd:element name="jobStepID" type="xsd:long"
default="0" />
            <xsd:element name="jobTaskID" type="xsd:long"
default="0" />
            <xsd:element name="agentID" type="xsd:string"
minOccurs="0" />
            <xsd:element name="queryBand" type="xsd:string"
minOccurs="0" />
            <xsd:element name="useQueryBandSource" type="xsd:boolean"
minOccurs="0" />
            <xsd:element name="useQueryBandTarget" type="xsd:boolean"
minOccurs="0" />
            <xsd:element name="lastUpdateTime" type="xsd:long"
minOccurs="0" />
            <xsd:element name="baseJobName" type="xsd:string"
minOccurs="0" />

```



```

        <xsd:element name="utility" type="xsd:string"
minOccurs="0" />
        <xsd:element name="srcSystem" type="xsd:string"
minOccurs="0" />
        <xsd:element name="targetSystem" type="xsd:string"
minOccurs="0" />
    </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="sqlType">
    <!-- Restricting the values to a set of value using 'enumeration'
-->
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="NONE" />
        <xsd:enumeration value="ADD_EXISTING_INDICES" />
        <xsd:enumeration value="ADD_SECONDARY_INDEX" />
        <xsd:enumeration value="ALTER_TRIGGER" />
        <xsd:enumeration value="COPY_STATS" />
        <xsd:enumeration value="CREATE_STAGING_TABLE" />
        <xsd:enumeration value="CREATE_TARGET_TABLE" />
        <xsd:enumeration value="CREATE_TRIGGER" />
        <xsd:enumeration value="DELETE_ALL" />
        <xsd:enumeration value="DELETE_MATCHING_ROWS" />
        <xsd:enumeration value="DELETE_PARTIAL_ROWS" />
        <xsd:enumeration value="DROP" />
        <xsd:enumeration value="DROP_PRIMARY_KEY_SI" />
        <xsd:enumeration value="DROP_SECONDARY_INDEX" />
        <xsd:enumeration value="DROP_STAGING_TABLE" />
        <xsd:enumeration value="DROP_TARGET_INDICES" />
        <xsd:enumeration value="HARD_DELETE" />
        <xsd:enumeration value="INSERT" />
        <xsd:enumeration value="INSERT_SELECT" />
        <xsd:enumeration value="RENAME_STAGING_TABLE" />
        <xsd:enumeration value="RENAME_TABLE" />
        <xsd:enumeration value="REVALIDATE_TABLE" />
        <xsd:enumeration value="SELECT" />
        <xsd:enumeration value="UPDATE" />
        <xsd:enumeration value="UPSERT" />
        <xsd:enumeration value="MERGE" />
        <xsd:enumeration value="DELETE_ARC_PARTIAL" />
        <xsd:enumeration value="CHANGE_DEFAULT_DATABASE" />
        <xsd:enumeration value="REDISTRIBUTE_JAR" />
        <xsd:enumeration value="CREATE_VIEW_DATA_TABLE" />
    </xsd:restriction>
</xsd:simpleType>

```

```

        <xsd:enumeration value="ASTER_MR_FUNCTION" />
        <xsd:enumeration value="ASTER_CREATE_TARGET_TABLE" />
        <xsd:enumeration
value="ASTER_CREATE_TARGET_TABLE_USING_SQLMR" />
        <xsd:enumeration value="ASTER_DROP_TARGET_TABLE"/>
        <xsd:enumeration value="ASTER_DROP_STAGING_TABLE"/>
        <xsd:enumeration value="ASTER_CREATE_STAGING_TABLE"/>
        <xsd:enumeration value="ASTER_CREATE_DIMENSION_TABLE"/>
        <xsd:enumeration value="ASTER_MERGE"/>
        <xsd:enumeration value="ASTER_DELETE_ALL"/>
        <xsd:enumeration value="ASTER_DELETE_MATCHING_ROWS"/>
        <xsd:enumeration value="ASTER_INSERT_SELECT"/>
        <xsd:enumeration value="ASTER_ADD_SI"/>
        <xsd:enumeration value="ASTER_CREATE_DRIVER_TABLE"/>
        <xsd:enumeration value="ASTER_DROP_DRIVER_TABLE"/>
        <xsd:enumeration value="DROP_TARGET_TABLE"/>
        <xsd:enumeration
value="RENAME_STAGING_TABLE_TO_TARGET_TABLE"/>
        <xsd:enumeration value="DIAGNOSTIC_IDCOL_TO_DEFAULT"/>
        <xsd:enumeration value="FOREIGN_SERVER_INSERT_SELECT"/>
        <xsd:enumeration value="CREATE_SOURCE_STAGING_TABLE"/>
        <xsd:enumeration value="MOVE_DATA_TO_SOURCE_STAGING"/>
        <xsd:enumeration value="DROP_SOURCE_STAGING_TABLE"/>
        <xsd:enumeration value="FOREIGN_SERVER_INSERT_SELECT_TARGET"/>
    </xsd:restriction>
</xsd:simpleType>

    <xsd:complexType name="sqlStatementType">
        <xsd:sequence>
            <xsd:element name="sql_text" minOccurs="0"
type="xsd:string" />
            <xsd:element name="job_task_ID" minOccurs="0"
type="xsd:long" />
            <xsd:element name="sequence_number" minOccurs="0"
type="xsd:int" />
            <xsd:element name="type" minOccurs="0" type="sqlType" />
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="sysCmdType">
        <xsd:sequence>
            <xsd:element name="cmd_action" type="xsd:string" />
            <xsd:element name="cmd_args" type="xsd:string" />
            <xsd:element name="blockMode" type="xsd:boolean" />

```

```

        </xsd:sequence>
    </xsd:complexType>

    <xsd:element name="dmLogStatus">
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="dmOutputBase">
                    <xsd:sequence>
                        <xsd:element name="jobTaskId"
type="xsd:long" />
                        <xsd:element name="type"
type="objectStatusEnumType" />
                        <xsd:element name="logLine"
type="xsd:string" />
                        <xsd:element name="logCode"
type="xsd:int" />
                        <xsd:element name="time"
type="xsd:string" />
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="dmJobTaskStatus">
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="dmOutputBase">
                    <xsd:sequence>
                        <xsd:element
name="job_task_status" type="taskStatusType" />
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="dmSyncBarrierMessage">
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="dmOutputBase">
                    <xsd:sequence>
                        <xsd:element
name="sync_barrier_message" type="syncBarrierMessageType" />
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>

```

```

        </xsd:sequence>
    </xsd:extension>
</xsd:complexContent>
</xsd:complexType>
</xsd:element>

<xsd:element name="dmSyncTelinfoMessage">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmOutputBase">
                <xsd:sequence>
                    <xsd:element
name="sync_telinfo_message" type="syncTelinfoMessageType" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<xsd:element name="dmSyncSchemaMessage">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmOutputBase">
                <xsd:sequence>
                    <xsd:element
name="sync_schema_message" type="syncSchemaMessageType" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<xsd:element name="dmAgentAbortMessage">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmOutputBase">
                <xsd:sequence>
                    <xsd:element name="jobTaskId"
type="xsd:long" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

```

```

    <xsd:element name="dmAgentRetrievedMacroNameMessage">
      <xsd:complexType>
        <xsd:complexContent>
          <xsd:extension base="dmOutputBase">
            <xsd:sequence>
              <xsd:element name="jobTaskId"
type="xsd:long" />
              <xsd:element name="macroName"
type="xsd:string" />
            </xsd:sequence>
          </xsd:extension>
        </xsd:complexContent>
      </xsd:complexType>
    </xsd:element>

    <xsd:element name="dmAgentRetrievedActualSessionsMessage">
      <xsd:complexType>
        <xsd:complexContent>
          <xsd:extension base="dmOutputBase">
            <xsd:sequence>
              <xsd:element name="jobTaskId"
type="xsd:long" />
              <xsd:element
name="actualSessions" type="xsd:string" />
              <xsd:element name="operatorType"
type="xsd:int" />
              <xsd:element name="instanceID"
type="xsd:int" />
            </xsd:sequence>
          </xsd:extension>
        </xsd:complexContent>
      </xsd:complexType>
    </xsd:element>

    <!-- Define triggers -->
    <xsd:element name="triggers">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="trigger" minOccurs="0"
maxOccurs="unbounded"
type="triggerType" />
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>

```

```

    </xsd:element>

    <xsd:complexType name="triggerType">
        <xsd:sequence>
            <!-- Qualifier database name for the trigger name -->
            <xsd:element name="database" nillable="false"
type="xsd:string" />

            <!-- Qualifier database name for the subject table -->
            <xsd:element name="subject_table_database"
nillable="false"
                type="xsd:string" />

            <!-- Name of the subject table to be associated with the
trigger -->
            <xsd:element name="table" nillable="false"
type="xsd:string" />

            <!-- Trigger name -->
            <xsd:element name="name" nillable="false"
type="xsd:string" />

            <!-- Trigger action time for trigger to be fired -->
            <xsd:element name="action_time" type="actionType"
                default="after" />
        </xsd:sequence>
        <xsd:attribute name="selection" type="selection"
use="required" />
    </xsd:complexType>

    <xsd:complexType name="actionType">
        <xsd:simpleContent>
            <xsd:extension base="actionTimeType">
                <xsd:attribute name="enabled" type="enabledFlag"
                    default="no" />
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>

    <!-- actionTimeType -->
    <xsd:simpleType name="actionTimeType">
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="[Bb][Ee][Ff][Oo][Rr][Ee]|[Aa][Ff][Tt]
[Ee][Rr]"/>

```

```

        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="enabledFlag">
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="[Yy][Ee][Ss]|[Nn][Oo]"/>
        </xsd:restriction>
    </xsd:simpleType>

    <!-- Define Join Index/Hash Index -->
    <xsd:element name="indices">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="index" minOccurs="0"
maxOccurs="unbounded"
                                type="joinIndexType" />
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:simpleType name="indexEnumType">
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="[Hh][Aa][Ss][Hh]_[Ii][Nn][Dd][Ee][Xx]|
[Jj][Oo][Ii][Nn]_[Ii][Nn][Dd][Ee][Xx]"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:complexType name="associateTableType">
        <xsd:sequence>
            <xsd:element name="database" type="xsd:string" />
            <xsd:element name="table" type="xsd:string" />
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="joinIndexType">
        <xsd:sequence>
            <!-- index name -->
            <xsd:element name="name" nillable="false"
type="xsd:string" />

            <!-- Database that owns the index -->
            <xsd:element name="target_name" type="xsd:string" maxOccurs="1"
minOccurs="0"></xsd:element>
            <xsd:element name="index_database" nillable="false"

```

```

        type="xsd:string" />
        <xsd:element name="map" minOccurs="0" maxOccurs="1"
type="xsd:string" />
        <xsd:element name="colocate" minOccurs="0" maxOccurs="1"
type="xsd:string" />
        <!-- index type -->
        <xsd:element name="target_index_database"
            type="targetDatabaseType" maxOccurs="1"
minOccurs="0">
            </xsd:element>
            <xsd:element name="index_type" nillable="false"
                type="indexEnumType" />

        <!-- associate table types -->
        <xsd:element name="associate_table" minOccurs="0"
            maxOccurs="unbounded"
type="associateTableType" />

    </xsd:sequence>
    <xsd:attribute name="copyStats" type="booleanType"
use="optional"/>
    <xsd:attribute name="selection" type="selection"
use="required" />

</xsd:complexType>

<!-- Define journals -->
<xsd:element name="journals">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="journal" minOccurs="0"
maxOccurs="unbounded"
                type="journalType" />
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:complexType name="journalType">
    <xsd:sequence>
        <!-- Database that owns the journal -->
        <xsd:element name="journal_database" nillable="false"
            type="xsd:string" />

```



```

        <!-- Journal name -->
        <xsd:element name="name" nillable="false"
type="xsd:string" />
    </xsd:sequence>
    <xsd:attribute name="selection" type="selection"
use="required" />
</xsd:complexType>

<!-- ##### Define View ##### -->
<xsd:element name="views">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="view" minOccurs="1"
maxOccurs="unbounded"
                                type="viewType" />
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:complexType name="viewType">
    <xsd:sequence>
        <!-- view name -->
        <xsd:element name="name" nillable="false"
type="xsd:string" />
        <!-- view database -->
        <xsd:element name="database" nillable="false"
type="xsd:string" />
        <xsd:element name="db_client_encryption" minOccurs="0"
type="triStateType" default="unspecified"/>
        <xsd:element name="view_data_table"
type="viewDataTableType" minOccurs="0" maxOccurs="1"/>

        <xsd:element name="use_source_staging_table"
minOccurs="0"
                                maxOccurs="1" type="booleanType"
default="false" />
        <xsd:element name="source_staging_database" type="targetDatabaseType"
maxOccurs="1" minOccurs="0" />
        <!-- target_staging_database new tag used to replace
staging_database -->
        <xsd:element name="target_staging_database"
type="targetDatabaseType" minOccurs="0" maxOccurs="1" />

        <xsd:element name="staging_database"

```

```

                                type="targetDatabaseType" minOccurs="0"
maxOccurs="1" />
                                <xsd:element name="staging_database_for_table"
                                type="targetDatabaseType" minOccurs="0"
maxOccurs="1" />
                                <xsd:element name="force_target_staging_table"
                                type="booleanType" minOccurs="0" maxOccurs="1" />
                                <xsd:element name="validate_row_count" minOccurs="0"
                                maxOccurs="1" type="RowCountValidationType"
default="none" />
                                <xsd:element name="compare_ddl" minOccurs="0"
maxOccurs="1"
                                type="triStateType" default="unspecified" />
                                <!-- partial table copy -->
                                <xsd:element name="sql_where_clause" minOccurs="0"
                                maxOccurs="1" type="xsd:string" />
                                <xsd:element ref="key_columns" minOccurs="0"
maxOccurs="1" />
                                </xsd:sequence>
                                <xsd:attribute name="selection" type="selection"
use="required" />
                                <xsd:attribute name="copyData" type="booleanType" use="optional"
                                default="false" />
                                </xsd:complexType>

                                <xsd:complexType name="viewDataTableType">
                                    <xsd:sequence>
                                        <xsd:element name="target_table" type="xsd:string"
minOccurs="1" maxOccurs="1"/>
                                        <xsd:element name="target_database" type="xsd:string"
minOccurs="1" maxOccurs="1"/>
                                    </xsd:sequence>
                                </xsd:complexType>
                                <!-- ##### -->

                                <!-- ##### Define Foreign_servers ##### -->
                                <xsd:element name="foreign_servers">
                                    <xsd:complexType>
                                        <xsd:sequence>
                                            <xsd:element name="foreign_server" minOccurs="1"
maxOccurs="unbounded"
                                            type="foreignServerType" />

```

```

        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:complexType name="foreignServerType">
    <xsd:sequence>
        <xsd:element name="name" type="xsd:string" minOccurs="1"
maxOccurs="1"/>
        <xsd:element name="map" minOccurs="0" maxOccurs="1"
type="xsd:string" />
        <xsd:element name="colocate" minOccurs="0" maxOccurs="1"
type="xsd:string" />
    </xsd:sequence>
    <xsd:attribute name="selection" type="selection"
use="required" />
</xsd:complexType>

<!-- ##### -->

<!-- ##### Define Function Aliases ##### -->

<xsd:element name="function_aliases">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="function_alias" minOccurs="1"
maxOccurs="unbounded"
                                type="functionAliasType" />
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:complexType name="functionAliasType">
    <xsd:sequence>
        <!-- function Alias name -->
        <xsd:element name="name" nillable="false"
type="xsd:string" />
        <!-- function Alias database name -->
        <xsd:element name="database" nillable="false"
type="xsd:string" />
    </xsd:sequence>
    <xsd:attribute name="selection" type="selection"
use="required" />
</xsd:complexType>
<!-- ##### -->

```

```

<!-- ##### Define TRACE_LOG ##### -->
<xsd:complexType name="traceLogType">
    <xsd:sequence>
        <xsd:element name="cli_trace_log" minOccurs="0"
maxOccurs="1"
                type="cliTraceLogType" />
        <xsd:element name="tpt_trace_log" minOccurs="0"
maxOccurs="1"
                type="tptTraceLogType" />
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="cliTraceLogType">
    <xsd:sequence>
        <xsd:element name="netrace" type="xsd:int" minOccurs="0"
maxOccurs="1"/>
        <xsd:element name="netrace_buf_len" type="xsd:int"
minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="tptTraceLogType">
    <xsd:sequence>
        <xsd:element name="tptapi_debug" type="xsd:int"
minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>
<!-- ##### -->

<!-- ##### Define Macro ##### -->
<xsd:element name="macros">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="macro" minOccurs="1"
maxOccurs="unbounded"
                    type="macroType" />
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:complexType name="macroType">
    <xsd:sequence>

```

```

        <!-- macro name -->
        <xsd:element name="name" nillable="false"
type="xsd:string" />
        <!-- macro database -->
        <xsd:element name="database" nillable="false"
type="xsd:string" />
    </xsd:sequence>
    <xsd:attribute name="selection" type="selection"
use="required" />
</xsd:complexType>

<!-- ##### Define Schema ##### -->
<xsd:element name="schemas">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="schema" minOccurs="1"
maxOccurs="unbounded"
                                type="schemaType" />
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:complexType name="schemaType">
    <xsd:sequence>
        <!-- schema name -->
        <xsd:element name="name" nillable="false"
type="xsd:string" />
        <xsd:element name="compare_ddl" minOccurs="0"
maxOccurs="1"
                                type="triStateType" default="unspecified" />
    </xsd:sequence>
    <xsd:attribute name="selection" type="selection"
use="required" />
</xsd:complexType>

<!-- ##### Define Stored Procedures ##### -->
<xsd:element name="stored_procedures">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="stored_procedure"
minOccurs="1" maxOccurs="unbounded"
                                type="storedProcedureType" />

```

```

        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:complexType name="storedProcedureType">
    <xsd:sequence>
        <!-- stored procedure name -->
        <xsd:element name="name" nillable="false"
type="xsd:string" />
        <!-- stored procedure database -->
        <xsd:element name="database" nillable="false"
type="xsd:string" />
    </xsd:sequence>
    <xsd:attribute name="selection" type="selection"
use="required" />
</xsd:complexType>

<!-- ##### Define Functions ##### -->
<xsd:element name="functions">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="function" minOccurs="1" maxOccurs="unbounded"
type="functionType" />
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:complexType name="functionType">
    <xsd:sequence>
        <!-- function name -->
        <xsd:element name="name" nillable="false" type="xsd:string" />
        <!-- function database -->
        <xsd:element name="database" nillable="false" type="xsd:string" />
    </xsd:sequence>
    <xsd:attribute name="selection" type="selection" use="required" />
</xsd:complexType>

<!--
    Define table with sql select statement with a WHERE clause for
partial
    table copy
-->
<xsd:complexType name="tableType">

```

```

        <xsd:sequence>
            <xsd:element name="name" minOccurs="1" maxOccurs="1"
                type="xsd:string" />
            <xsd:element name="owner_name" minOccurs="0"
maxOccurs="1"
                type="xsd:string" />

            <xsd:element name="db_client_encryption" minOccurs="0"
type="triStateType" default="unspecified"/>

            <!-- option to use staging database -->
            <xsd:element name="use_source_staging_table"
minOccurs="0"
                maxOccurs="1" type="booleanType"
default="false" />
            <xsd:element name="source_staging_database" type="targetDatabaseType"
maxOccurs="1" minOccurs="0" />
            <!-- target_staging_database new tag used to replace
staging_database -->
            <xsd:element name="target_staging_database"
type="targetDatabaseType" minOccurs="0" maxOccurs="1" />

            <xsd:element name="staging_database"
                type="targetDatabaseType" minOccurs="0"
maxOccurs="1" />
            <xsd:element name="staging_database_for_table"
                type="targetDatabaseType" minOccurs="0"
maxOccurs="1" />
            <xsd:element name="force_target_staging_table"
                type="booleanType" minOccurs="0" maxOccurs="1" />
            <xsd:element name="target_database"
                type="targetDatabaseType" maxOccurs="1"
minOccurs="0" />
            <xsd:element name="target_name" type="xsd:string"
                maxOccurs="1" minOccurs="0">
        </xsd:element>
        <!-- Aster specific parameters -->
        <xsd:element name="aster_table_params" minOccurs="0"
maxOccurs="1" type="asterTableParamsType" />
        <!-- validate row count -->
        <xsd:element name="validate_row_count" minOccurs="0"
                maxOccurs="1" type="RowCountValidationType"
default="none" />

        <!-- option to override lock access -->

```

```

        <xsd:element name="override_lock_access" minOccurs="0"
            maxOccurs="1" type="booleanType"
default="false" />
        <!-- use export without spool feature when using TPTAPI --
>
        <xsd:element name="export_without_spool" minOccurs="0"
            maxOccurs="1" type="triStateType"
default="unspecified" />
        <!-- compare DDL of the source table with the target table
-->
        <xsd:element name="compare_ddl" minOccurs="0"
maxOccurs="1"
            type="triStateType" default="unspecified" />
        <xsd:element name="map" minOccurs="0" maxOccurs="1"
type="xsd:string" />
        <xsd:element name="colocate" minOccurs="0" maxOccurs="1"
type="xsd:string" />
        <!-- compare journaling forward if table uses journaling
-->
        <xsd:element name="journaling" minOccurs="0"
maxOccurs="1"
            type="triStateType" default="unspecified" />
        <!-- partial table copy -->
        <xsd:element name="sql_where_clause" minOccurs="0"
            maxOccurs="1" type="xsd:string" />
        <xsd:element ref="key_columns" minOccurs="0"
maxOccurs="1" />
        <xsd:element name="staging_to_target"
            type="StagingToTargetType" maxOccurs="1"
minOccurs="0"/>
        <xsd:element name="table_columns"
            type="tableColumnsType" minOccurs="0"
maxOccurs="1" />
    </xsd:sequence>
    <xsd:attribute name="copyStats" type="booleanType"
use="optional"/>
    <xsd:attribute name="selection" type="selection"
use="required" />
    <xsd:attribute name="allowTptLoadForMultiset" type="booleanType"
default="false" use="optional" />
</xsd:complexType>

<!-- Aster Table parameters -->
<xsd:complexType name="asterTableParamsType">

```



```

        <xsd:sequence>
            <xsd:element name="source_schema_name" minOccurs="0"
maxOccurs="1" type="xsd:string" />
            <xsd:element name="target_schema_name" minOccurs="0"
maxOccurs="1" type="xsd:string" />
            <xsd:element name="target_table_type" minOccurs="0"
maxOccurs="1" type="asterTableType"/>
            <xsd:element name="target_table_distribution_type"
minOccurs="0" maxOccurs="1" type="asterTableDistributionType"/>
            <xsd:element name="target_table_distribution_key_column"
minOccurs="0" maxOccurs="1" type="xsd:string" />
        </xsd:sequence>
    </xsd:complexType>
<!-- The target table (aster) should be either FACT or DIMENSION, by default it's
FACT table. -->
    <xsd:simpleType name="asterTableType">
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="[Ff][Aa][Cc][Tt]|[Dd][Ii][Mm][Ee][Nn]
[Ss][Ii][Oo][Nn]"/>
        </xsd:restriction>
    </xsd:simpleType>
<!-- The target table (aster) distribution type should be either HASH or
REPLICATION, FACT table should be HASH and DIMENSION table can be either. -->
    <xsd:simpleType name="asterTableDistributionType">
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="[Dd][Ii][Ss][Tt][Rr][Ii][Bb][Uu][Tt]
[Ee][_][Bb][Yy][_][Hh][Aa][Ss][Hh]|[Dd][Ii][Ss][Tt][Rr][Ii][Bb][Uu][Tt][Ee][_]
[Bb][Yy][_][Rr][Ee][Pp][Ll][Ii][Cc][Aa][Tt][Ii][Oo][Nn]" />
        </xsd:restriction>
    </xsd:simpleType>

    <!-- Define key_columns -->
    <xsd:element name="key_columns">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="key_column" minOccurs="1"
maxOccurs="unbounded"
                                type="xsd:string" />
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:simpleType name="charsetType">
        <xsd:restriction base="xsd:string">

```

```

        <xsd:enumeration value="ASCII" />
        <xsd:enumeration value="UTF8" />
        <xsd:enumeration value="UTF16" />
    </xsd:restriction>
</xsd:simpleType>

<!-- Basic information about a job -->
<xsd:simpleType name="jobCopyType">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="master" />
        <xsd:enumeration value="instance" />
    </xsd:restriction>
</xsd:simpleType>

```

```

<xsd:complexType name="simpleJobType">
    <xsd:sequence>
        <!-- job name -->
        <xsd:element name="name" type="xsd:string" />
        <!-- Job Cleanup status and results -->
        <xsd:element name="cleanupPerformed"
            type="xsd:boolean"
            maxOccurs="1" minOccurs="0"/>
        <xsd:element name="cleanup_message" type="xsd:string"
            minOccurs="0" />
        <xsd:element name="cleanup_result" type="xsd:string"
            maxOccurs="unbounded"
            minOccurs="0" />
    </xsd:sequence>
    <!-- master copy or instance copy -->
    <xsd:attribute name="copy" type="jobCopyType"
        use="required" />
</xsd:complexType>

<xsd:complexType name="targetDatabaseType">
    <xsd:sequence>
        <xsd:element name="name" type="xsd:string" maxOccurs="1"
            minOccurs="1"></xsd:element>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="useForeignServerType">
    <xsd:sequence>
        <xsd:element name="name" type="xsd:string" maxOccurs="1"
            minOccurs="1"></xsd:element>
    </xsd:sequence>
</xsd:complexType>

```

```

        </xsd:sequence>
        <xsd:attribute name="foreignServerOnTarget" type="triStateType"
default="unspecified" use="optional"/>
    </xsd:complexType>

    <xsd:element name="dmCompareDDLTask">
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="dmAgentCommandBase">
                    <xsd:sequence>
                        <xsd:element name="source_tdpid"
                            type="xsd:string" />
                        <xsd:element
name="source_username"
                            type="xsd:string" />
                        <xsd:element
name="source_logon_mech"
                            type="xsd:string"
maxOccurs="1" minOccurs="0"/>
                        <xsd:element
name="source_logon_mech_data"
                            type="xsd:string"
maxOccurs="1" minOccurs="0"/>
                        <xsd:element
name="source_account_id"
                            type="xsd:string"
maxOccurs="1" minOccurs="0"/>
                        <xsd:element name="target_tdpid"
                            type="xsd:string" />
                        <xsd:element
name="target_username"
                            type="xsd:string" />
                        <xsd:element
name="target_logon_mech"
                            type="xsd:string"
maxOccurs="1" minOccurs="0"/>
                        <xsd:element
name="target_logon_mech_data"
                            type="xsd:string"
maxOccurs="1" minOccurs="0"/>
                        <xsd:element
name="target_account_id"
                            type="xsd:string"
maxOccurs="1" minOccurs="0"/>

```

```

        <xsd:element name="useBaseViews"
            type="xsd:boolean" />
        <xsd:element
name="performance_report_rate"
            type="xsd:int"
default="0" />
        <xsd:element
name="session_charset"
            type="xsd:string"
maxOccurs="1" minOccurs="0">
            </xsd:element>
            <xsd:element name="jobTask" type="jobTaskType">
                </xsd:element>
                <xsd:element name="compare_ddl_tables"
type="compareDDLTableType" maxOccurs="unbounded" minOccurs="1" />
            <xsd:element
name="data_encrypted_by_agent" nillable="true"
                maxOccurs="unbounded"
type="dmAgentEncryptedBase" />
            <xsd:element
name="source_session_charset" minOccurs="0" type="xsd:string"></xsd:element>
            <xsd:element
name="target_session_charset" minOccurs="0" type="xsd:string"></xsd:element>
            <xsd:element
name="database_client_encryption" type="xsd:boolean" default="false" />
            <xsd:element
name="source_jdbc_connection_string"
type="xsd:string" maxOccurs="1" minOccurs="0"/>
            <xsd:element
name="target_jdbc_connection_string"
type="xsd:string" maxOccurs="1" minOccurs="0"/>
        </xsd:sequence>
    </xsd:extension>
</xsd:complexContent>
</xsd:complexType>
</xsd:element>

    <xsd:element name="dmAsterCompareDDLTask">
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="dmAgentCommandBase">

```

```

<xsd:sequence>

    <xsd:element
name="source_db_type" minOccurs="0"

                                type="xsd:string" />
    <xsd:element
name="source_db_name" minOccurs="0"

                                type="xsd:string" />
    <xsd:element
name="source_db_port" minOccurs="0"

                                type="xsd:string" />

    <xsd:element name="source_tdpid"
                                type="xsd:string"

maxOccurs="1" minOccurs="0"/>
    <xsd:element
name="source_username"

                                type="xsd:string" />
    <xsd:element
name="source_logon_mech"

                                type="xsd:string"

maxOccurs="1" minOccurs="0"/>
    <xsd:element
name="source_logon_mech_data"

                                type="xsd:string"

maxOccurs="1" minOccurs="0"/>
    <xsd:element
name="source_account_id"

                                type="xsd:string"

maxOccurs="1" minOccurs="0"/>

    <xsd:element
name="target_db_type" minOccurs="0"

                                type="xsd:string" />
    <xsd:element
name="target_db_name" minOccurs="0"

                                type="xsd:string" />
    <xsd:element
name="target_db_port" minOccurs="0"

                                type="xsd:string" />

    <xsd:element name="target_tdpid"

```

```

                                type="xsd:string"
maxOccurs="1" minOccurs="0"/>
                                <xsd:element
                                type="xsd:string" />
                                <xsd:element
                                type="xsd:string"
                                <xsd:element
                                type="xsd:string"
                                <xsd:element
                                type="xsd:string"
                                <xsd:element name="useBaseViews"
                                type="xsd:boolean"
                                <xsd:element
                                type="xsd:int"
                                <xsd:element
                                type="xsd:string"
                                </xsd:element>
                                <xsd:element name="jobTask" type="jobTaskType">
                                </xsd:element>
                                <xsd:element name="compare_ddl_tables"
                                type="asterCompareDDLTableType" maxOccurs="unbounded" minOccurs="1" />
                                <xsd:element
                                name="data_encrypted_by_agent" nillable="true"
                                maxOccurs="unbounded"
                                type="dmAgentEncryptedBase" />
                                <xsd:element
                                name="source_session_charset" minOccurs="0" type="xsd:string"></xsd:element>
                                <xsd:element
                                name="target_session_charset" minOccurs="0" type="xsd:string"></xsd:element>
                                <xsd:element name="compare_ddl"
                                type="triStateType" default="unspecified" />

```

```

                                <xsd:element
name="database_client_encryption" type="xsd:boolean" default="false" />
                                </xsd:sequence>
                                </xsd:extension>
                                </xsd:complexContent>
                                </xsd:complexType>
                                </xsd:element>

                                <xsd:complexType name="compareDDLTableType">
                                    <xsd:sequence>
                                        <xsd:element name="src_name" type="xsd:string"></
xsd:element>
                                        <xsd:element name="src_database" type="xsd:string"></
xsd:element>
                                        <xsd:element name="target_name" type="xsd:string"
maxOccurs="1" minOccurs="0"></xsd:element>
                                        <xsd:element name="target_database" type="xsd:string"
maxOccurs="1" minOccurs="0"></xsd:element>
                                        <xsd:element name="selected" type="xsd:boolean"></
xsd:element>
                                        <xsd:element name="objectType" type="xsd:string"></
xsd:element>
                                    </xsd:sequence>
                                </xsd:complexType>

                                <xsd:complexType name="asterCompareDDLTableType">
                                    <xsd:sequence>
                                        <xsd:element name="selection" type="xsd:boolean" />
                                        <xsd:element name="src_name" type="xsd:string" />
                                        <xsd:element name="src_schema" type="xsd:string"
minOccurs="0" maxOccurs="1" />
                                        <xsd:element name="src_database" type="xsd:string"
minOccurs="0" maxOccurs="1" />
                                        <xsd:element name="target_name" type="xsd:string"
minOccurs="0" maxOccurs="1" />
                                        <xsd:element name="target_schema" type="xsd:string"
minOccurs="0" maxOccurs="1" />
                                        <xsd:element name="target_database" type="xsd:string"
minOccurs="0" maxOccurs="1" />
                                        <xsd:element name="exists_on_target"
type="xsd:boolean" />
                                        <xsd:element name="compare_ddl" type="triStateType"
default="unspecified" />
                                        <xsd:element name="table_columns"

```

```

                                type="tableColumnsType" minOccurs="0"
maxOccurs="1" />
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="dmAgentEncryptedBase">
        <xsd:sequence>
            <xsd:element name="agent_public_key" minOccurs="1"
                maxOccurs="1" type="xsd:string" />
            <xsd:element name="encrypted_symmetric_key" minOccurs="1"
                maxOccurs="1" type="xsd:string" />
        </xsd:sequence>
    </xsd:complexType>

    <!--
        Data Encrypted by Agent public key, source and target password are
required
    -->
    <xsd:complexType name="dmAgentEncryptedTPTJDBCTask">
        <xsd:complexContent>
            <xsd:extension base="dmAgentEncryptedBase">
                <xsd:sequence>
                    <xsd:element
name="source_password_encrypted"
                                minOccurs="1"
                                type="xsd:string" />
                    <xsd:element
name="target_password_encrypted"
                                minOccurs="1"
                                type="xsd:string" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <!-- Data Encrypted by Agent public key, target password is optional -->
    <xsd:complexType name="dmAgentEncryptedSQLTask">
        <xsd:complexContent>
            <xsd:extension base="dmAgentEncryptedBase">
                <xsd:sequence>
                    <xsd:element
name="target_password_encrypted"
                                minOccurs="0"
                                type="xsd:string" />

```



```

                                <xsd:element
name="source_password_encrypted"
                                minOccurs="0"
type="xsd:string" />
                                </xsd:sequence>
                                </xsd:extension>
                                </xsd:complexContent>
                                </xsd:complexType>

<!-- Data Encrypted by Agent public key, target password is optional -->
<xsd:complexType name="dmAgentEncryptedAsterMRTask">
  <xsd:complexContent>
    <xsd:extension base="dmAgentEncryptedBase">
      <xsd:sequence>
        <xsd:element
name="source_password_encrypted"
                                minOccurs="0"
type="xsd:string" />
                                <xsd:element
name="target_password_encrypted"
                                minOccurs="0"
type="xsd:string" />
                                </xsd:sequence>
                                </xsd:extension>
                                </xsd:complexContent>
                                </xsd:complexType>

```

```

<!--
  Data Encrypted by Agent public key, source and target password are
optional
-->
<xsd:complexType name="dmAgentEncryptedCompareDDLTask">
  <xsd:complexContent>
    <xsd:extension base="dmAgentEncryptedBase">
      <xsd:sequence>
        <xsd:element
name="source_password_encrypted"
                                minOccurs="0"
type="xsd:string" />
                                <xsd:element
name="target_password_encrypted"
                                minOccurs="0"
type="xsd:string" />
                                </xsd:sequence>

```

```

        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<!-- dmModifyAdminPassword -->
<xsd:element name="dmModifyAdminPassword">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmCommandBase">
                <xsd:sequence>
                    <xsd:element
name="current_password" minOccurs="0"
                                type="xsd:string" />
                    <xsd:element
name="current_password_encrypted"
                                minOccurs="0"
type="xsd:string" />
                    <xsd:element name="new_password"
                                type="xsd:string" />
                    <xsd:element
name="new_password_encrypted"
                                minOccurs="0"
type="xsd:string" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<!-- dmModifyAdminPasswordOutput -->
<xsd:element name="dmModifyAdminPasswordOutput">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmOutputBase"/>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<!-- dmCheckSecurityType -->
<xsd:complexType name="dmCheckSecurityType">
    <xsd:sequence>
        <xsd:element name="security_username" minOccurs="0"
type="xsd:string" />
    </xsd:sequence>
</xsd:complexType>

```

```

        <xsd:element name="security_password" minOccurs="0"
type="xsd:string" />
        <xsd:element name="security_password_encrypted"
minOccurs="0" type="xsd:string" />
        <xsd:element name="security_role" minOccurs="0"
maxOccurs="unbounded" type="xsd:string" />
        <xsd:element name="security_portlet_token" minOccurs="0"
type="xsd:string" />
        <xsd:element name="security_jwt" minOccurs="0"
maxOccurs="1" type="dmJWTDataType" />
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="dmJWTDataType">
    <xsd:sequence>
        <xsd:element name="jwt" type="xsd:string" minOccurs="1"
maxOccurs="1" />
        <xsd:element name="encrypted_symmetric_key"
type="xsd:string" minOccurs="1" maxOccurs="1" />
    </xsd:sequence>
</xsd:complexType>

<xsd:element name="dmJobListUpdateAnnouncement">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmMessageBase">
                <xsd:sequence>
                    <xsd:element name="timestamp"
type="xsd:string" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<xsd:element name="dmJobDefinitionUpdateAnnouncement">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmMessageBase">
                <xsd:sequence>
                    <xsd:element name="jobName"
minOccurs="1" type="xsd:string"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

```

```

        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<xsd:element name="dmJobStatusUpdateAnnouncement">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmMessageBase">
                <xsd:sequence>
                    <xsd:element name="jobName"
minOccurs="1" type="xsd:string"/>
                    <xsd:element name="jobComplete"
minOccurs="1" type="xsd:boolean"/>
                    <xsd:element
name="blockedJobStarted" minOccurs="1" type="xsd:boolean"/>
                    <xsd:element name="objectName"
minOccurs="0" type="xsd:string"/>
                    <xsd:element
name="totalBytesProcessed" minOccurs="0" type="xsd:long"/>
                    <xsd:element name="jobExecution"
type="jobExecutionType" maxOccurs="1" minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<!-- dmEditJob, edit command used by portlet -->
<xsd:element name="dmEditJob">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="jobDefinitionType">
                <xsd:sequence>
                    <xsd:element
name="variableChanged" type="jobVariablesChangeType"
minOccurs="0"
maxOccurs="1" />
                    <xsd:element name="job_security"
minOccurs="0" type="securityType" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

```

```

<!-- dmEditJobOutput -->
<xsd:element name="dmEditJobOutput">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="dmOutputBase"/>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<!-- dmEdit, edit command used by commandline, all the fields are optional
-->
<xsd:element name="dmEdit">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="changedJobDefinitionType">
        <xsd:sequence>
          <xsd:element name="job_security"
minOccurs="0" type="securityType" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<!-- dmEditOutput -->
<xsd:element name="dmEditOutput">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="dmOutputBase"/>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<!-- Get Daemon Current Time-->
<xsd:element name="dmDaemonTime">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="dmCommandBase">
        <xsd:sequence>
          <xsd:element
name="response_timeout" minOccurs="0"
                                type="xsd:int" default="10" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

```

```

        </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<xsd:element name="dmDaemonTimeOutput">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmOutputBase">
                <xsd:sequence>
                    <xsd:element name="daemon_time"
type="xsd:dateTime"
                                minOccurs="1"
maxOccurs="1" />
                                <xsd:element
name="daemon_time_zone_label" type="xsd:string"
                                minOccurs="1"
maxOccurs="1" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<xsd:element name="dmDaemonPingError">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmOutputBase">
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>

<xsd:element name="dmPingPortlet">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmCommandBase">
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>

<xsd:element name="dmPortletDataTransferMessage">

```

```

        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="dmMessageBase">
                    <xsd:sequence>
                        <xsd:element
name="upgradeVersion" type="xsd:string" />
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="dmPortletDataTransferResult">
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="dmOutputBase">
                    <xsd:sequence>
                        <xsd:element
name="upgradeVersion" type="xsd:string" />
                        <xsd:element
name="portletDataTrasfer" type="dmPortletDaemonDataUpgradeInfo" minOccurs="0"
maxOccurs="unbounded" />
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>

    <xsd:complexType name="dmPortletDaemonDataUpgradeInfo">
        <xsd:sequence>
            <xsd:element name="nick" type="xsd:string" />
            <xsd:element name="code" type="xsd:string" />
            <xsd:element name="data" type="xsd:string"
minOccurs="0"
maxOccurs="unbounded" />
        </xsd:sequence>
    </xsd:complexType>

    <xsd:element name="dmUpdateJobPriorities">
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="dmCommandBase">
                    <xsd:sequence>

```

```

                                <xsd:element
name="job_priority_set" minOccurs="1" maxOccurs="unbounded"
type="updateJobPriorityType" />
                                </xsd:sequence>
                                </xsd:extension>
                                </xsd:complexContent>
                                </xsd:complexType>
                                </xsd:element>

                                <xsd:element name="dmUpdateJobPrioritiesOutput">
                                <xsd:complexType>
                                <xsd:complexContent>
                                <xsd:extension base="dmOutputBase">
                                <xsd:sequence>
                                <xsd:element name="success_list"
minOccurs="1" maxOccurs="1" type="jobListType" />
                                <xsd:element name="failed_list"
minOccurs="1" maxOccurs="1" type="jobListWithReasonType" />
                                </xsd:sequence>
                                </xsd:extension>
                                </xsd:complexContent>
                                </xsd:complexType>
                                </xsd:element>

                                <xsd:complexType name="updateJobPriorityType">
                                <xsd:sequence>
                                <xsd:element name="job_name" minOccurs="1" maxOccurs="1"
type="xsd:string" />
                                <xsd:element name="job_priority" minOccurs="1"
maxOccurs="1" type="jobPriorityType" />
                                </xsd:sequence>
                                </xsd:complexType>

                                <xsd:complexType name="jobListType">
                                <xsd:sequence>
                                <xsd:element name="job_name" minOccurs="0"
maxOccurs="unbounded" type="xsd:string" />
                                </xsd:sequence>
                                </xsd:complexType>

                                <xsd:complexType name="jobListWithReasonType">
                                <xsd:sequence>
                                <xsd:element name="failed_job" minOccurs="0"
maxOccurs="unbounded" type="failedJobType" />

```



```

        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="failedJobType">
        <xsd:sequence>
            <xsd:element name="job_name" minOccurs="1" maxOccurs="1"
type="xsd:string" />
            <xsd:element name="failed_reason" minOccurs="1"
maxOccurs="1" type="xsd:string" />
        </xsd:sequence>
    </xsd:complexType>

    <xsd:simpleType name="logonMechanismType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="default" />
            <xsd:enumeration value="kerberos" />
        </xsd:restriction>
    </xsd:simpleType>

    <!-- Aster system details -->
    <xsd:complexType name="asterSystemType">
        <xsd:sequence>
            <xsd:element name="system_name" minOccurs="1"
maxOccurs="1" type="xsd:string" />
            <xsd:element name="port" minOccurs="0" maxOccurs="1"
type="xsd:string" />
            <xsd:element name="user_name" minOccurs="1" maxOccurs="1"
type="xsd:string" />
            <xsd:element name="password" minOccurs="0" maxOccurs="1"
type="xsd:string" />
            <xsd:element name="password_encrypted" minOccurs="0"
maxOccurs="1" type="xsd:string" />
        </xsd:sequence>
    </xsd:complexType>

    <!-- asterDefinitionType # Aster Job, list of parameters required for
Aster specific job -->
    <xsd:complexType name="asterOptionType">
        <xsd:sequence>
            <xsd:element name="query_timeout" minOccurs="0"
maxOccurs="1" type="xsd:int" />
            <xsd:element name="preserve_column_case" minOccurs="0"
maxOccurs="1" type="enabledFlag" />

```

```

        <xsd:element name="skip_error_records" minOccurs="0"
maxOccurs="1" type="enabledFlag" />
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="dsaOptionsType">
    <xsd:sequence>
        <xsd:element name="target_group_name" minOccurs="0" maxOccurs="1"
type="xsd:string" />
        <xsd:element name="parallel_builds" minOccurs="0" maxOccurs="1"
type="xsd:int" default="5" />
        <xsd:element name="ir_allow_write" minOccurs="0"
maxOccurs="1" type="triStateType" default="unspecified" />
        <xsd:element name="ir_execution_type" minOccurs="0"
maxOccurs="1" type="irExecutionType" default="unspecified"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="irExecutionType">
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="[Ff][Uu][Ll][Ll]|[Uu][Nn][Ss][Pp][Ee]
[Cc][Ii][Ff][Ii][Ee][Dd]"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="tableColumnsType">
    <xsd:sequence>
        <xsd:element name="column" minOccurs="1" maxOccurs="unbounded"
type="tableColumnType" />
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="tableColumnType">
    <xsd:sequence>
        <xsd:element name="name" minOccurs="1" maxOccurs="1"
type="xsd:string" />
        <xsd:element name="target_name" minOccurs="0"
maxOccurs="1" type="xsd:string" />
        <xsd:element name="type" minOccurs="0" maxOccurs="1"
type="xsd:string" />
        <xsd:element name="target_type" minOccurs="0"
maxOccurs="1" type="xsd:string" />
        <xsd:element name="size" minOccurs="0" maxOccurs="1"
type="xsd:string" />
    </xsd:sequence>
</xsd:complexType>

```

```

        <xsd:element name="charset" minOccurs="0" maxOccurs="1"
type="xsd:string" />
        <xsd:element name="allowNull" minOccurs="0" maxOccurs="1"
type="triStateType" default="unspecified" />
        <xsd:element name="allowDuplicate" minOccurs="0"
maxOccurs="1" type="triStateType" default="unspecified" />
        <xsd:element name="isPrimaryIndex" minOccurs="0"
maxOccurs="1" type="triStateType" default="unspecified" />
    </xsd:sequence>
</xsd:complexType>

<!--
    Define StatsTask, which holds stats metadata (job_task_stats
metadata table) for Multi Column stats, Column stats and Index stats.
-->
<xsd:complexType name="statsTaskType">
    <xsd:sequence>
        <xsd:element name="table_name" minOccurs="1"
maxOccurs="1" type="xsd:string" />
        <xsd:element name="owner_name" minOccurs="1"
maxOccurs="1" type="xsd:string" />
        <xsd:element name="target_name" minOccurs="1"
maxOccurs="1" type="xsd:string" />
        <xsd:element name="target_owner_name" minOccurs="1"
maxOccurs="1" type="xsd:string" />
        <xsd:element name="stats_type" minOccurs="0"
maxOccurs="1" type="xsd:string" />
        <xsd:element name="stats_id" minOccurs="0" maxOccurs="1"
type="xsd:string" />
        <xsd:element name="index_name" minOccurs="0"
maxOccurs="1" type="xsd:string" />
        <xsd:element name="index_type" minOccurs="0"
maxOccurs="1" type="xsd:string" />
        <xsd:element name="unique_flag" minOccurs="0"
maxOccurs="1" type="xsd:boolean" />
        <xsd:element name="column_name" minOccurs="0"
maxOccurs="1" type="xsd:string" />
        <xsd:element name="column_type" minOccurs="0"
maxOccurs="1" type="xsd:string" />
        <xsd:element name="column_format" minOccurs="0"
maxOccurs="1" type="xsd:string" />
        <xsd:element name="column_length" minOccurs="0"
maxOccurs="1" type="xsd:string" />
        <xsd:element name="decimal_total_digits" minOccurs="0"

```

```

maxOccurs="1" type="xsd:int" />
        <xsd:element name="decimal_fractional_digits"
minOccurs="0" maxOccurs="1" type="xsd:int" />
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="s3PropertiesType">
    <xsd:sequence>
        <xsd:element name="access_key_id" minOccurs="1" maxOccurs="1"
type="xsd:string"/>
        <xsd:element name="secret_access_key" minOccurs="0" maxOccurs="1"
type="xsd:string"/>
        <xsd:element name="secret_access_key_encrypted" minOccurs="0"
maxOccurs="1" type="xsd:string"/>
        <xsd:element name="buckets_by_regions" minOccurs="1" maxOccurs="1"
type="bucketsByRegionsType" />
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="bucketsByRegionsType">
    <xsd:sequence>
        <xsd:element name="buckets_by_region" minOccurs="1"
maxOccurs="unbounded" type="bucketsByRegionType"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="bucketsByRegionType">
    <xsd:sequence>
        <xsd:element name="region" minOccurs="1" maxOccurs="1"
type="xsd:string"/>
        <xsd:element name="buckets" minOccurs="1" maxOccurs="1">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="bucket" minOccurs="1"
maxOccurs="unbounded" type="cloudStagingBucketType" />
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="cloudStagingBucketType">
    <xsd:sequence>
        <xsd:element name="bucket_name" minOccurs="1" maxOccurs="1"

```

```

type="xsd:string"/>
    <xsd:element name="prefix_list" minOccurs="1" maxOccurs="1"
type="prefixListType"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="prefixListType">
    <xsd:sequence>
        <xsd:element name="prefix" minOccurs="1" maxOccurs="unbounded"
type="prefixType"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="prefixType">
    <xsd:sequence>
        <xsd:element name="prefix_name" minOccurs="1" maxOccurs="1"
type="xsd:string"/>
        <xsd:element name="storage_devices" minOccurs="1" maxOccurs="1"
type="xsd:long"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="sourceTargetPairType">
    <xsd:sequence>
        <xsd:element name="source_system" minOccurs="1"
maxOccurs="1" type="xsd:string"/>
        <xsd:element name="source_system_target_group" minOccurs="0"
maxOccurs="1" type="xsd:string"/>
        <xsd:element name="target_system" minOccurs="1"
maxOccurs="1" type="xsd:string"/>
        <xsd:element name="target_system_target_group" minOccurs="0"
maxOccurs="1" type="xsd:string" />
        <xsd:element name="valid" minOccurs="0" maxOccurs="1"
type="xsd:boolean" />
        <xsd:element name="note" minOccurs="0" maxOccurs="1"
type="xsd:string" />
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="cloudStagingType">
    <xsd:complexContent>
        <!-- This extension contains the required security content defined in
the common section of the document -->
        <xsd:extension base="dmCommandBase">

```

```

        <xsd:sequence>
            <xsd:element name="name" minOccurs="1" maxOccurs="1"
type="xsd:string"/>
            <xsd:element name="storage_type" minOccurs="1" maxOccurs="1"
type="xsd:string" default="S3" />
            <xsd:element name="s3_properties" minOccurs="0" maxOccurs="1"
type="s3PropertiesType" />
            <xsd:element name="source_target_pairs" minOccurs="1"
maxOccurs="1">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="source_target_pair" minOccurs="1"
maxOccurs="unbounded" type="sourceTargetPairType" />
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
    </xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:element name="dmCreateCloudStaging">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="cloudStagingType" />
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<xsd:element name="dmEditCloudStaging">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="cloudStagingType" />
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<xsd:complexType name="deleteCloudStagingType">
    <xsd:complexContent>
        <xsd:extension base="dmCommandBase">
            <xsd:sequence>
                <xsd:element name="name" minOccurs="1" maxOccurs="1"
type="xsd:string"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

```

```

        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="dmDeleteCloudStaging">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="deleteCloudStagingType" />
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<xsd:complexType name="listCloudStagingAreasType">
    <xsd:complexContent>
        <xsd:extension base="dmCommandBase" />
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="dmListCloudStagingAreas">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="listCloudStagingAreasType" />
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>

<xsd:complexType name="getCloudStagingAreaType">
    <xsd:complexContent>
        <!-- This extension contains the required security content defined in
the common section of the document -->
        <xsd:extension base="dmCommandBase">
            <xsd:sequence>
                <xsd:element name="name" minOccurs="1" maxOccurs="1"
type="xsd:string"/>
                <xsd:element name="filename" minOccurs="0" maxOccurs="1"
type="xsd:string"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:element name="dmGetCloudStagingArea">
    <xsd:complexType>
        <xsd:complexContent>

```

```

        <xsd:extension base="getCloudStagingAreaType" />
    </xsd:complexContent>
</xsd:complexType>
</xsd:element>

<xsd:element name="dmCloudStagingOutput">
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="dmOutputBase">
                <xsd:sequence>
                    <xsd:element name="httpStatusCode" nillable="false"
type="xsd:int"/>
                    <xsd:element name="requestOutput" nillable="false"
type="xsd:string" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>
</xsd:schema>

```


Additional Information

Audience

This guide is intended for use by:

- Database administrators
- System administrators
- Software developers, production users, and testers

The following prerequisite knowledge is required for this product:

- Dual-active systems
- Teradata Database
- Teradata system hardware

Changes and Additions

Date	Release	Description
September 2022	17.20.01.00	Defined the capability of Data Mover commands to backup and restore Cloud Staging Copy Service repository.
June 2022	17.20.00.00	<ul style="list-style-type: none"> • Added information about how you can choose to use a foreign server for a T2T job from source system and target system. • Added the capabilities of Cloud Staging Copy Service in Data Mover. • Introduced the cloud staging area, with configuration, usage and restrictions. • Added how to create cloud staging area using Data Mover portlet. • Added new command parameters and RESTful APIs intended for Cloud Staging Copy Service. • Updated the Data Mover XML schemas.
January 2022	17.12.00.00	<ul style="list-style-type: none"> • Added information about moving individual views, macros, stored procedures, and copying functions using DSA. • Added the column description of the TMSMEVENT table.
September 2021	17.11.00.00	<ul style="list-style-type: none"> • Added DSA Incremental Copy Job • Updated Running a Job under Job Management • Added Single sign-on support • Added Ignore Missing Source Objects • Updated information for backup_daemon command
June 2021	17.10.00.00	Added support for TLS 1.2 with SQL-Engine

Date	Release	Description
March 2021	17.06.00.00	Initial Release – Limited Availability Only

Teradata Links

Link	Description
https://docs.teradata.com/	Search Teradata Documentation, customize content to your needs, and download PDFs. Customers: Log in to access Orange Books.
https://support.teradata.com	Helpful resources in one place: <ul style="list-style-type: none"> • Support requests • Account management and software downloads • Knowledge base, community, and support policies • Product documentation • Learning resources, including Teradata University
https://www.teradata.com/University/Overview	Teradata education network
https://support.teradata.com/community	Link to Teradata community

Related Documentation

Title	Publication ID
<i>Teradata® Data Mover Installation, Configuration, and Upgrade Guide for Customers</i> Describes how to configure Teradata Data Mover software.	B035-4102
<i>Teradata® Viewpoint User Guide</i> Describes the Teradata Viewpoint portal, portlets, and system administration features.	B035-2206
<i>Teradata® Ecosystem Manager User Guide</i> Describes how to use Teradata Ecosystem Manager portlets.	B035-3201
<i>Teradata® Database JSON Data Type</i> Describes Teradata support for JSON data.	B035-1150

Customer Education

Teradata Customer Education delivers training for your global workforce, including scheduled public courses, customized on-site training, and web-based training. For information about the classes, schedules, and the Teradata Certification Program, go to <https://www.teradata.com/TEN/>.

Customer Support

Customer support is available around-the-clock, seven days a week through the Global Technical Support Center (GSC). To learn more, go to <https://support.teradata.com>.