# Tungsten Replicator 7.1 Manual

Continuent Ltd

# Tungsten Replicator 7.1 Manual

Continuent Ltd


Abstract

This manual documents Tungsten Replicator 7.1. This includes information for:

- Tungsten Replicator

*Build date:* 2024-05-13 (e4ffb2e8)

Up to date builds of this document: Tungsten Replicator 7.1 Manual (Online), Tungsten Replicator 7.1 Manual (PDF)

# Table of Contents

# List of Figures

# List of Tables

# Preface

This manual documents Tungsten Replicator 7.1 up to and including 7.1.2 build 81. Differences between minor versions are highlighted stating the explicit minor release version, such as 7.1.2.x.

For other versions and products, please use the appropriate manual.

## 1. Legal Notice



## 2. Conventions

This documentation uses a number of text and style conventions to indicate and differentiate between different types of information:

- *Text in this style* is used to show an important element or piece of information. It may be used and combined with other text styles as appropriate to the context.
- Text in this style is used to show a section heading, table heading, or particularly important emphasis of some kind.
- Program or configuration options are formatted using `this style`. Options are also automatically linked to their respective documentation page when this is known. For example, tpm and `--hosts` [409] both link automatically to the corresponding reference page.
- Parameters or information explicitly used to set values to commands or options is formatted using *`this style`*.
- Option values, for example on the command-line are marked up using this format: `--help`. Where possible, all option values are directly linked to the reference information for that option.
- Commands, including sub-commands to a command-line tool are formatted using Text in this style. Commands are also automatically linked to their respective documentation page when this is known. For example, tpm links automatically to the corresponding reference page.
- `Text in this style` indicates literal or character sequence text used to show a specific value.
- Filenames, directories or paths are shown like this `/etc/passwd`. Filenames and paths are automatically linked to the corresponding reference page if available.

Bulleted lists are used to show lists, or detailed information for a list of items. Where this information is optional, a magnifying glass symbol enables you to expand, or collapse, the detailed instructions.

Code listings are used to show sample programs, code, configuration files and other elements. These can include both user input and replaceable values:

```
shell> cd /opt/continuent/software
shell> ar zxvf tungsten-replicator-7.1.2-81.tar.gz
```

In the above example command-lines to be entered into a shell are prefixed using `shell`. This shell is typically sh, ksh, or bash on Linux and Unix platforms.

If commands are to be executed using administrator privileges, each line will be prefixed with root-shell, for example:

```
root-shell> vi /etc/passwd
```

To make the selection of text easier for copy/pasting, ignorable text, such as `shell>` are ignored during selection. This allows multi-line instructions to be copied without modification, for example:

```
mysql> create database test_selection;
mysql> drop database test_selection;
```

Lines prefixed with `mysql>` should be entered within the mysql command-line.

If a command-line or program listing entry contains lines that are two wide to be displayed within the documentation, they are marked using the » character:

```
the first line has been extended by using a »
    continuation line
```

They should be adjusted to be entered on a single line.

Text marked up with **`this style`** is information that is entered by the user (as opposed to generated by the system). Text formatted using *`this style`* should be replaced with the appropriate file, version number or other variable information according to the operation being performed.

In the HTML versions of the manual, blocks or examples that can be userinput can be easily copied from the program listing. Where there are multiple entries or steps, use the 'Show copy-friendly text' link at the end of each section. This provides a copy of all the user-enterable text.

# 3. Quickstart Guide

- Are you planning on completing your first installation?
  - Have you followed the Appendix B, *Prerequisites*?
  - Have you chosen your deployment type? See Chapter 2, *Deployment Overview*
  - Is this a Primary/Replica deployment?
  - Are you looking to configure an applier??
- Are you using the Tungsten Replicator AMI available in the Amazon AWS Marketplace?
- Would you like to understand the different types of installation?

  There are two installation methods available in tpm, INI and Staging. A comparison of the two methods is at

  Section 9.1, “Comparing Staging and INI tpm Methods”.
- Do you want to upgrade to the latest version?

  See Section 7.14.1, “ Upgrading Tungsten Replicator using tpm ”.
- Are you trying to update or change the configuration of your system?

  See Section 9.5.20, “tpm update Command”.
- Would you like to perform database or operating system maintenance?

  See Section 7.13, “Performing Database or OS Maintenance”.
- Do you need to backup or restore your system?

  For backup instructions, see Section 7.7, “Creating a Backup”, and to restore a previously made backup, see Section 7.8, “Restoring a Backup”.

# Chapter 1. Introduction

Tungsten Replicator™ is a replication engine supporting a variety of different extractor and applier modules. Data can be extracted from MySQL, Amazon RDS MySQL, Amazon Aurora, Microsoft Azure and Google Cloud SQL, and applied to a variety of transactional stores, NoSQL stores and datawarehouse stores. For a full list of supported sources and targets, see Table 1.1, "Supported Extractors" and Table 1.2, "Supported Appliers" below

During replication, Tungsten Replicator assigns data a unique global transaction ID, and enables flexible statement and/or row-based replication of data. This enables data to be exchanged between different databases and different database versions. During replication, information can be filtered and modified, and deployment can be between on-premise or cloud-based databases. For performance, Tungsten Replicator™ provides support for parallel replication, and advanced topologies such as fan-in, star and active/active, and can be used efficiently in cross-site deployments.

Tungsten Replicator™ is the core foundation for Tungsten Cluster™ for HA, DR and geographically distributed solutions.

Features in Tungsten Replicator

- Includes support for replicating into Hadoop (including Apache Hadoop, Cloudera, HortonWorks, MapR, Amazon EMR)
- Includes support for replicating into Amazon Redshift, including storing change data within Amazon S3
- Includes support for replicating into PostgreSQL, Apache Kafka, MongoDB
- Includes support for replicating to and from Amazon Aurora/RDS (MySQL) deployments
- Available as an AMI via Amazon Marketplace (Without Support)
- SSL Support for managing MySQL deployments
- Network Client filter for handling complex data translation/migration needs during replication

The table below shows the version of Tungsten Replicator that support was added for the specific extractor

### Table 1.1. Supported Extractors

| Source | 5.3 | 5.4 | 6.0 | 6.1 | 7.0 |
|---|---|---|---|---|---|
| MySQL (5.0 to 5.6) | x | x | x | x | x |
| MySQL 5.7 | x | x | x | x | x |
| MySQL 8 | | x | | x | x |
| MariaDB (5.5, 10) | x | x | x | x | x |
| Amazon Aurora/RDS MySQL | x | x | x | x | x |
| Google Cloud MySQL | x | x | x | x | x |
| Microsoft Azure | x | x | x | x | x |

The table below shows the version of Tungsten Replicator that support was added for the specific applier

### Table 1.2. Supported Appliers

| Target | 5.3 | 5.4 | 6.0 | 6.1 | 7.0 |
|---|---|---|---|---|---|
| MySQL (incl MariaDB) | x | x | x | x | x |
| Amazon Aurora/RDS MySQL | x | x | x | x | x |
| Microsoft Azure | x | x | x | x | x |
| Google Cloud MySQL | x | x | x | x | x |
| Oracle (incl. Cloud) | x | x | x | x | x |
| PostgreSQL (incl. Cloud) | x | x | x | x | x |
| Hadoop | x | x | x | x | x |
| Vertica | x | x | x | x | x |
| Amazon Redshift | x | x | x | x | x |
| MongoDB | x | x | x | x | x |
| MongoDB Atlas | | | | x (6.1.3) | x |

| Target | 5.3 | 5.4 | 6.0 | 6.1 | 7.0 |
|---|---|---|---|---|---|
| Apache Kafka | x | x | x | x | x |
| Clickhouse | | | | x | x |

# 1.1. Tungsten Replicator

Tungsten Replicator is a high performance replication engine that works with a number of different source and target databases to provide high-performance and improved replication functionality over the native solution. With MySQL replication, for example, the enhanced functionality and information provided by Tungsten Replicator allows for global transaction IDs, advanced topology support such as Composite Active/Active, star, and fan-in, and enhanced latency identification.

In addition to providing enhanced functionality Tungsten Replicator is also capable of heterogeneous replication by enabling the replicated information to be transformed after it has been read from the data server to match the functionality or structure in the target server. This functionality allows for replication between MySQL and a variety of heterogeneous targets.

Understanding how Tungsten Replicator works requires looking at the overall replicator structure. There are three major components in the system that provide the core of the replication functionality:

- Extractor

  The extractor component reads data from a MysQL data server and writes that information into the Transaction History Log (THL). The role of the extractor is to read the information from a suitable source of change information and write it into the THL in the native or defined format, either as SQL statements or row-based information.

  Information is always extracted from a source database and recorded within the THL in the form of a complete transaction. The full transaction information is recorded and logged against a single, unique, transaction ID used internally within the replicator to identify the data.

- Applier

  Appliers within Tungsten Replicator convert the THL information and apply it to a destination data server. The role of the applier is to read the THL information and apply that to the data server.

  The applier works with a number of different target databases, and is responsible for writing the information to the database. Because the transactional data in the THL is stored either as SQL statements or row-based information, the applier has the flexibility to reformat the information to match the target data server. Row-based data can be reconstructed to match different database formats, for example, converting row-based information into an Oracle-specific table row, or a MongoDB document.

- Transaction History Log (THL)

  The THL contains the information extracted from a data server. Information within the THL is divided up by transactions, either implied or explicit, based on the data extracted from the data server. The THL structure, format, and content provides a significant proportion of the functionality and operational flexibility within Tungsten Replicator.

  As the THL data is stored additional information, such as the metadata and options in place when the statement or row data was extracted are recorded. Each transaction is also recorded with an incremental global transaction ID. This ID enables individual transactions within the THL to be identified, for example to retrieve their content, or to determine whether different appliers within a replication topology have written a specific transaction to a data server.

These components will be examined in more detail as different aspects of the system are described with respect to the different systems, features, and functionality that each system provides.

From this basic overview and structure of Tungsten Replicator, the replicator allows for a number of different topologies and solutions that replicate information between different services. Straightforward replication topologies, such as Primary/Replica are easy to understand with the basic concepts described above. More complex topologies use the same core components. For example, Composite Active/Active topologies make use of the global transaction ID to prevent the same statement or row data being applied to a data server multiple times. Fan-in topologies allow the data from multiple data servers to be combined into one data server.

## 1.1.1. Extractor

Extractors exist for reading information from the following sources:

- Reading the MySQL binary log (binlog) directly from the disk and translating that content and session information into the THL. Using this method to read the binlog in it's different formats, such as the statement, row and mixed-based logging.

- Remotely from MySQL server over a network, including reading from an Amazon RDS MySQL or Amazon Aurora instance. This enables the replicator to read the information remotely, either on services where direct access to the binlog is not available, or where we cannot be installed (Such as databases hosted on a Windows platform).

## 1.1.2. Appliers

Once information has been recorded into THL, particularly when that information has been recorded in row-based format, it is possible to apply that information out to a variety of different targets, both transactional and SQL based solutions, and also NoSQL and analytical targets.

Available appliers include:

- MySQL
  - Community Edition
  - Enterprise Edition
  - Percona
  - MariaDB
  - Amazon Aurora/RDS (Including cross region)
  - Google Cloud SQL
  - Microsoft Azure
- Oracle
- PostgreSQL
- Amazon RedShift
- HPE Vertica
- Hadoop, compatible with all major distributions
- MongoDB (Including Atlas from v6.1.3 onwards)
- Apache Kafka
- Clickhouse (Experimental)

For more information on how the heterogeneous replicator works, see Section 2.8.1, "How Heterogeneous Replication Works". For more information on the batch applier, which works with datawarehouse targets, see Section 5.6, "Batch Loading for Data Warehouses".

## 1.1.3. Transaction History Log (THL)

Tungsten Replicator operates by reading information from the source database and transferring that information to the *Transaction History Log (THL)*.

Each transaction within the THL includes the SQL statement or the row-based data written to the database. The information also includes, where possible, transaction specific options and metadata, such as character set data, SQL modes and other information that may affect how the information is written when the data is applied. The combination of the metadata and the global transaction ID also enable more complex data replication scenarios to be supported, such as Composite Active/Active, without fear of duplicating statement or row data application because the source and global transaction ID can be compared.

In addition to all this information, the THL also includes a timestamp and a record of when the information was written into the database before the change was extracted. Using a combination of the global transaction ID and this timing information provides information on the latency and how up to date a dataserver is compared to the original datasource.

Depending on the underlying storage of the data, the information can be reformatted and applied to different data servers. When dealing with row-based data, this can be applied to a different type of data server, or completely reformatted and applied to non-table based services such as MongoDB.

THL information is stored for each replicator service, and can also be exchanged over the network between different replicator instances. This enables transaction data to be exchanged between different hosts within the same network or across wide-area-networks.

## 1.1.4. Filtering

Filtering within the replicator enables the information within the THL to be removed, augmented, or modified as the information is transferred within and between the replicators.

During filtering, the information in the THL can be modified in a host of different ways, including but not limited to:

- Filtering out information based on the schema name, table name or column name. This is useful if you want a subset of the information in your target database, or if you want want to apply only certain columns to the information.
- Filter information based on the content, or value of one or more fields.
- Filter information based on the operation type, for example, only applying inserts to a target ignoring updates or deletes.
- Modify or alter the format or structure of the data. This can be used to change the data format to be compatible with a target system, for example due to data type limitations, or sizes.
- Add information to the data. For example, adding a database name, source name, or additional or compound fields into the target data. Within an analytics system this can be useful when combining data from multiple sources so that the source system or customer can still be identified.

The format, content, and structure of the data and the THL can be modified and new data can even be created through the filters.

For more information on the filters available, and how to use them, see Chapter 11, *Replication Filters*.

# Chapter 2. Deployment Overview

Tungsten Replicator creates a unique replication interface between two databases. Because Tungsten Replicator is independent of the dataserver it affords a number of different advantages, including more flexible replication strategies, filtering, and easier control to pause, restart, and skip statements between hosts.

Replication is supported from, and to, different dataservers using different technologies through a series of extractor and applier components which independently read data from, and write data to, the dataservers in question.

The replication process is made possible by reading the binary log on each host. The information from the binary log is written into the Tungsten Replicator Transaction History Log (THL), and the THL is then transferred between hosts and then applied to each Target host. More information can be found in Chapter 1, *Introduction*.

Before covering the basics of creating different dataservices, there are some key terms that will be used throughout the setup and installation process that identify different components of the system. these are summarised in Table 2.1, "Key Terminology".

Table 2.1. Key Terminology

| Tungsten Term | Traditional Term | Description |
|---|---|---|
| *dataserver* | Database | The database on a host. Datasources include MySQL, or Oracle. |
| *datasource* | Host or Node | One member of a dataservice and the associated Tungsten components. |
| *staging host* | - | The machine (and directory) from which Tungsten Replicator is installed and configured. The machine does not need to be the same as any of the existing hosts in the cluster. |
| *staging directory* | - | The directory where the installation files are located and the installer is executed. Further configuration and updates must be performed from this directory. |

Before attempting installation, there are a number of prerequisite tasks which must be completed to set up your hosts, database, and Tungsten Replicator service:

1. Setup a staging host from which you will configure and manage your installation.
2. Configure each host that will be used within your dataservice.
3. Configure your MySQL installation, so that Tungsten Replicator can work with the database.
4. Prepare and configure the target environment

The following sections provide guidance and instructions for creating a number of different deployment scenarios using Tungsten Replicator.

## 2.1. Deployment Sources

Tungsten Replicator is available in a number of different distribution types, and the methods for configuration available for these different packages differs. See Section 9.1, "Comparing Staging and INI tpm Methods" for more information on the available installation methods.

| Deployment Type/Package | TAR/GZip | RPM |
|---|---|---|
| Staging Installation | Yes | No |
| INI File Configuration | Yes | Yes |
| Deploy Entire Cluster | Yes | No |
| Deploy Per Machine | Yes | Yes |

Two primary deployment sources are available:

- Tar/GZip

  Using the TAR/GZip package creates a local directory that enables you to perform installs and updates from the extracted 'staging' directory, or use the INI file format.

- RPM Packages

  Using the RPM package format is more suited to using the INI file format, as hosts can be installed and upgraded to the latest RPM package independently of each other.

All packages are named according to the product, version number, build release and extension. For example:

```
tungsten-replicator-7.1.2-81.tar.gz
```

The version number is `7.1.2` and build number `81`. Build numbers indicate which build a particular release version is based on, and may be useful when installing patches provided by support.

## 2.1.1. Using the TAR/GZipped files

To use the TAR/GZipped packages, download the files to your machine and unpack them:

```
shell> cd /opt/continuent/software
shell> tar zxf tungsten-replicator-7.1.2-81.tar.gz
```

This will create a directory matching the downloaded package name, version, and build number from which you can perform an install using either the INI file or command-line configuration. To use, you will need to use the tpm command within the `tools` directory of the extracted package:

```
shell> cd tungsten-replicator-7.1.2-81
```

## 2.1.2. Using the RPM package files

The RPM packages can be used for installation, but are primarily designed to be in combination with the INI configuration file.

Installation

Installing the RPM package will do the following:

1. Create the `tungsten` system user if it doesn't exist
2. Make the `tungsten` system user part of the `mysql` group if it exists
3. Create the `/opt/continuent/software` directory
4. Unpack the software into `/opt/continuent/software`
5. Define the `$CONTINUENT_PROFILES` and `$REPLICATOR_PROFILES` environment variables
6. Update the profile script to include the `/opt/continuent/share/env.sh` script
7. Create the `/etc/tungsten` directory
8. Run tpm install if the `/etc/tungsten.ini` or `/etc/tungsten/tungsten.ini` file exists

Although the RPM packages complete a number of the pre-requisite steps required to configure your cluster, there are additional steps, such as configuring ssh, that you still need to complete. For more information, see Appendix B, *Prerequisites*.

By using the package files you are able to setup a new server by creating the `/etc/tungsten.ini` file and then installing the package. Any output from the tpm command will go to `/opt/continuent/service_logs/rpm.output`.

### Note

If you download the package files directly, you may need to add the signing key to your environment before the package will load properly.

For yum platforms (RHEL/CentOS/Amazon Linux), the rpm command is used :

```
root-shell> rpm --import http://www.continuent.com/RPM-GPG-KEY-continuent
```

For Ubuntu/Debian platforms, the gpg command is used :

```
root-shell> gpg --keyserver keyserver.ubuntu.com --recv-key 7206c924
```

Once an INI file has been created and the packages are available, the installation can be completed using:

- On RHEL/CentOS/Amazon Linux:

```
root-shell> yum install tungsten-replicator
```

- On Ubuntu/Debian:

```
root-shell> apt-get install tungsten-replicator
```

Upgrades

If you upgrade to a new version of the RPM package it will do the following:

1. Unpack the software into `/opt/continuent/software`

2. Run tpm update if the `/etc/tungsten.ini` or `/etc/tungsten/tungsten.ini` file exists

The tpm update will restart all Continuent Tungsten services so you do not need to do anything after upgrading the package file.

## 2.2. Best Practices

A successful deployment depends on being mindful during deployment, operations and ongoing maintenance.

### 2.2.1. Best Practices: Deployment

- Identify the best deployment method for your environment and use that in production and testing. See Section 9.1, "Comparing Staging and `INI` tpm Methods".
- Standardize the OS and database prerequisites. There are Ansible modules available for immediate use within AWS, or as a template for modifications.

  More information on the Ansible method is available in this blog article.
- Ensure that the output of the `hostname`command and the nodename entries in the Tungsten configuration match exactly prior to installing Tungsten.

  The configuration keys that define nodenames are: `--slaves` [423], `--dataservice-slaves` [423], `--members` [413], `--master` [412], `--dataservice-master-host` [412], `--masters` [412] and `--relay` [412]
- For security purposes you should ensure that you secure the following areas of your deployment:
  - Ensure that you create a unique installation and deployment user, such as tungsten, and set the correct file permissions on installed directories. See Section B.3.3, "Directory Locations and Configuration".
  - When using ssh and/or SSL, ensure that the ssh key or certificates are suitably protected. See Section B.3.2.2, "SSH Configuration".
  - Use a firewall, such as iptables to protect the network ports that you need to use. The best solution is to ensure that only known hosts can connect to the required ports for Tungsten Cluster. For more information on the network ports required for Tungsten Cluster operation, see Section B.3.2.1, "Network Ports".
  - If possible, use authentication and SSL connectivity between hosts to protext your data and authorisation for the tools used in your deployment.

    See Chapter 6, *Deployment: Security* for more information.
- Choose your topology from the deployment section and verify the configuration matches the basic settings. Additional settings may be included for custom features but the basics are needed to ensure proper operation. If your configuration is not listed or does not match our documented settings; we cannot guarantee correct operation.
- If you are using `ROW` replication, any triggers that run additional `INSERT`/`UPDATE`/`DELETE` operations must be updated so they do not run on the Replica servers.
- Make sure you know the structure of the Tungsten Cluster home directory and how to initialize your environment for administration. See Section 7.1, "The Home Directory" and Section 7.2, "Establishing the Shell Environment".
- Prior to migrating applications to Tungsten Cluster test failover and recovery procedures from Chapter 7, *Operations Guide*. Be sure to try recovering a failed Primary and reprovisioning failed Replicas.
- When deciding on the Service Name for your configurations, keep them simple and short and only use alphanumerics (Aa-Zz,0-9) and underscores (_).

### 2.2.2. Best Practices: Upgrade

In this section we identify the best practices for performing a Tungsten Software upgrade.

- Identify the deployment method chosen for your environment, Staging or INI. See Section 9.1, "Comparing Staging and `INI` tpm Methods".
- The best practice for Tungsten software is to upgrade All-at-Once, performing zero Primary switches.
- The Staging deployment method automatically does an All-at-Once upgrade - this is the basic design of the Staging method.
- For an INI upgrade, there are two possible ways, One-at-a-Time (with at least one Primary switch), and All-at-Once (no switches at all).
- See Section 9.4.3, "Upgrades with an INI File" for more information.
- Here is the sequence of events for a proper Tungsten upgrade on a 3-node cluster with the INI deployment method:

- Login to the Customer Downloads Portal and get the latest version of the software.
- Copy the file (i.e. `tungsten-clustering-7.0.2-161.tar.gz`) to each host that runs a Tungsten component.
- Set the cluster to policy `MAINTENANCE`
- On every host:
  - Extract the tarball under /opt/continuent/software/ (i.e. create `/opt/continuent/software/tungsten-clustering-7.0.2-161`)
  - cd to the newly extracted directory
  - Run the Tungsten Package Manager tool, tools/tpm update --replace-release
- For example, here are the steps in order:

```
On ONE database node:
shell> cctrl
cctrl> set policy maintenance
cctrl> exit

On EVERY Tungsten host at the same time:
shell> cd /opt/continuent/software
shell> tar xvzf tungsten-clustering-7.0.2-161.tar.gz
shell> cd tungsten-clustering-7.0.2-161

To perform the upgrade and restart the Connectors gracefully at the same time:
shell> tools/tpm update --replace-release

To perform the upgrade and delay the restart of the Connectors to a later time:
shell> tools/tpm update --replace-release --no-connectors
When it is time for the Connector to be promoted to the new version, perhaps after taking it out of the load balancer:
shell> tpm promote-connector

When all nodes are done, on ONE database node:
shell> cctrl
cctrl> set policy automatic
cctrl> exit
```

WHY is it ok to upgrade and restart everything all at once?

Let's look at each component to examine what happens during the upgrade, starting with the Manager layer.

Once the cluster is in Maintenance mode, the Managers cease to make changes to the cluster, and therefore Connectors will not reroute traffic either.

Since Manager control of the cluster is passive in Maintenance mode, it is safe to stop and restart all Managers - there will be zero impact to the cluster operations.

The Replicators function independently of client MySQL requests (which come through the Connectors and go to the MySQL database server), so even if the Replicators are stopped and restarted, there should be only a small window of delay while the replicas catch up with the Primary once upgraded. If the Connectors are reading from the Replicas, they may briefly get stale data if not using SmartScale.

Finally, when the Connectors are upgraded they must be restarted so the new version can take over. As discussed in this blog post, Zero-Downtime Upgrades, the Tungsten Cluster software upgrade process will do two key things to help keep traffic flowing during the Connector upgrade promote step:

- Execute `connector graceful-stop 30`to gracefully drain existing connections and prevent new connections.
- Using the new software version, initiate the start/retry feature which launches a new connector process while another one is still bound to the server socket. The new Connector process will wait for the socket to become available by retrying binding every 200ms by default (which is tunable), drastically reducing the window for application connection failures.

## 2.2.3. Best Practices: Operations

- Setup proper monitoring for all servers as described in Section 7.15, "Monitoring Tungsten Cluster".
- Configure the Tungsten Cluster services to startup and shutdown along with the server. See Section 2.5, "Configuring Startup on Boot".

## 2.2.4. Best Practices: Maintenance

- Your license allows for a testing cluster. Deploy a cluster that matches your production cluster and test all operations and maintenance operations there.

- Disable any automatic operating system patching processes. The use of automatic patching will cause issues when all database servers automatically restart without coordination. See Section 7.13.3, "Performing Maintenance on an Entire Dataservice".
- Regularly check for maintenance releases and upgrade your environment. Every version includes stability and usability fixes to ease the administrative process.

## 2.3. Common tpm Options During Deployment

There are a variety of tpm options that can be used to alter some aspect of the deployment during configuration. Although they might not be provided within the example deployments, they may be used or required for different installation environments. These include options such as altering the ports used by different components, or the commands and utilities used to monitor or manage the installation once deployment has been completed. Some of the most common options are included within this section.

Changes to the configuration should be made with tpm update. This continues the procedure of using tpm install during installation. See Section 9.5.20, "tpm update Command" for more information on using tpm update.

- `--datasource-systemctl-service` [402]

  On some platforms and environments the command used to manage and control the MySQL or MariaDB service is handled by a tool other than the services or /etc/init.d/mysql commands.

  Depending on the system or environment other commands using the same basic structure may be used. For example, within CentOS 7, the command is systemctl. You can explicitly set the command to be used by using the `--datasource-systemctl-service` [402] to specify the name of the tool.

  The format of the corresponding command that will be used is expected to follow the same format as previous commands, for example to start the database service::

  ```
  shell> systemctl mysql stop
  ```

  Different commands must follow the same basic structure, the command configured by `--datasource-systemctl-service` [402], the servicename, and the status (i.e. `stop`).

## 2.4. Starting and Stopping Tungsten Replicator

To shutdown a running Tungsten Replicator operation you must switch off the replicator:

```
shell> replicator stop
Stopping Tungsten Replicator Service...
Stopped Tungsten Replicator Service.
```

### Note

> Stopping the replicator in this way results in an ungraceful shutdown of the replicator. To perform a graceful shutdown, use trepctl offline first, then stop or restart the replicator.

To start the replicator service if it is not already running:

```
shell> replicator start
Starting Tungsten Replicator Service...
```

To restart the replicator (stop and start) service if it is not already running:

```
shell> replicator restart
Stopping Tungsten Replicator Service...
Stopped Tungsten Replicator Service.
Starting Tungsten Replicator Service...
```

For some scenarios, such as initiating a load within a heterogeneous environment, the replicator can be started up in the `OFFLINE` [195] state:

```
shell> replicator start offline
```

In a clustered environment, if the cluster was configured with `auto-enable=false` [395] then you will need to put each node online individually.

## 2.5. Configuring Startup on Boot

By default, Tungsten Replicator does not start automatically on boot. To enable Tungsten Replicator to start at boot time on a system supporting the Linux Standard Base (LSB), use the deployall script provided in the installation directory to create the necessary boot scripts on your system:

```
shell> sudo deployall
```

To disable automatic startup at boot time, use the undeployall command:

```
shell> sudo undeployall
```

# 2.6. Removing Datasources from a Deployment

Removing components from a dataservice is quite straightforward, usually involves both modifying the running service and changing the configuration. Changing the configuration is necessary to ensure that the host is not re-configured and installed when the installation is next updated.

In this section:

- Section 2.6.1, "Removing a Datasource from an Existing Deployment"

## 2.6.1. Removing a Datasource from an Existing Deployment

To remove a datasource from an existing deployment there are two primary stages, removing it from the active service, and then removing it from the active configuration.

For example, to remove `host6` from a service:

1. Login to host6.
2. Stop the replicator:

   ```
   shell> replicator stop
   ```

Now the node has been removed from the active dataservice, the host must be removed from the configuration.

1. Now you must remove the node from the configuration, although the exact method depends on which installation method used with tpm:
   - If you are using staging directory method with tpm:
     a. Change to the staging directory. The current staging directory can be located using tpm query staging:

     ```
     shell> tpm query staging
     tungsten@host1:/home/tungsten/tungsten-replicator-7.1.2-81
     shell> cd /home/tungsten/tungsten-replicator-7.1.2-81
     ```

     b. Update the configuration, omitting the host from the list of members of the dataservice:

     ```
     shell> tpm update alpha \
         --members=host1,host2,host3
     ```

   - If you are using the INI file method with tpm:
     - Remove the INI configuration file:

     ```
     shell> rm /etc/tungsten/tungsten.ini
     ```

2. Remove the installed software directory:

   ```
   shell> rm -rf /opt/continuent
   ```

# 2.7. Understanding Deployment Styles and Topologies

The following sections provide understanding around the different styles of deployment available and the different topologies that can be configured using Tungsten Replicator

## 2.7.1. Tungsten Replicator Extraction Operation

| Replication Operation Support | |
|---|---|
| Statements Replicated | Yes, within MySQL/MySQL Topologies only |
| Rows Replicated | Yes |
| Schema Replicated | Yes, within MySQL/MySQL Topologies only |
| ddlscan Supported | Yes, supported for mixed MySQL, and data warehouse targets |

Tungsten Replicator for MySQL operates by

- Reading the MySQL binary log (binlog) directly from the disk and translating that content and session information into the THL. Using this method to read the binlog in it's different formats, such as the statement, row and mixed-based logging.
- Remotely from the MySQL server over a network, including reading from an Amazon Aurora MySQL instance, for example. This enables the replicator to read the information remotely, either on services where direct access to the binlog is not available, or where we cannot be installed. This is also referred to as Offboard installation

The following diagrams show these two methods of extraction

Figure 2.1. Internals: MySQL Extraction

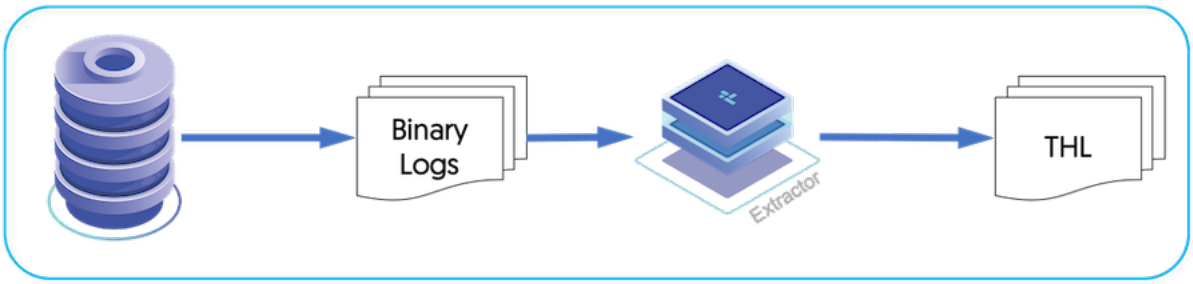
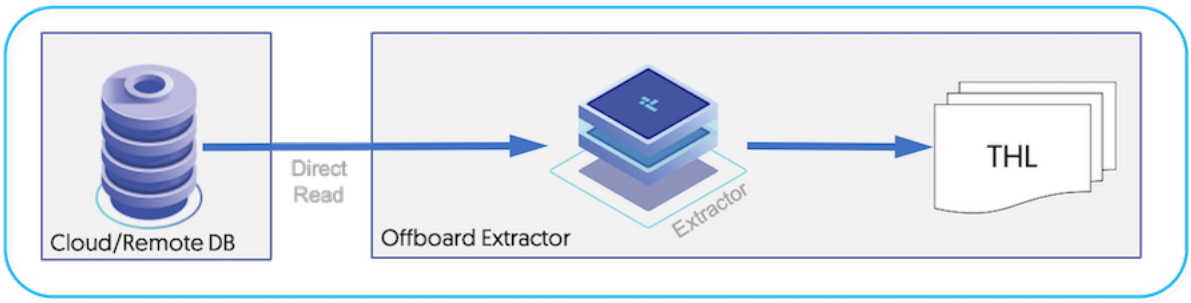



Figure 2.2. Internals: Amazon Aurora/Remote Database, Offboard Extraction




Tungsten Replicator for MySQL is supported within the following environments:

- MySQL Community Edition
- MySQL Enterprise Edition from Oracle
- Percona
- MariaDB
- Amazon RDS
- Amazon Aurora
- Google Cloud MySQL

In addition, the following requirements and limitations are in effect:

- Tables must have primary keys (Only applicable when the target is not Oracle, MySQL or Postgres)
- Row-based binary logging must be configured for heterogeneous deployment models
- Datatype support varies, depending upon the target. Check applier documentation appropriate to deployment target for more detail.
- Currently, DDL is only replicated in MySQL to MySQL deployments

## 2.7.2. Understanding Deployment Models

The flexibility of the replicator allows you to install the software in a number of ways to fit into a number of possible limitations or restrictions you may be faced with, in addition to a number of flexible topologies. These are outlined below

- Onboard

  This method will involve the Tungsten Replicator being installed on the same host as the Source MySQL Database. This method is suitable for:

  - On-Premise deployments
  - EC2 Hosted Databases in AWS
  - Google Cloud SQL Hosted Instances

- Offboard

  This method will involve the Tungsten Replicator being installed on the different host to the Source MySQL Database. This method is suitable for:

  - On-Premise deployments
  - EC2 Instances in AWS
  - Google Cloud SQL Hosted Instances
  - Amazon RDS MySQL Instances
  - Amazon Aurora Instances

- Direct

  This method involved the Tungsten Replicator being installed on a different host to the source MySQL Database, however the replicator will also act as the applier, writing out to the target This method is suitable for:

  - Amazon RDS MySQL Instances
  - Amazon Aurora Instances
  - Cluster-Extractor topologies, extracting direct from a Tungsten Cluster

- AWS Marketplace AMI

  This method is based on a pre-built AMI available for purchase within the Amazon Marketplace. This method is suitable for:

  - Amazon AWS Hosted solutions, including RDS and Aurora

## 2.7.3. Understanding Deployment Topologies

There are a number of different methods in which Tungsten Replicator can be configured, review Section 2.7.2, "Understanding Deployment Models" for full details of the differences between each deployment style. The following sections explain the different topology styles that can be deployed

- Section 2.7.3.1, "Simple Primary/Replica Topology"

  A simple Primary/Replica topology replicating from one source host to one target.

- Section 2.7.3.2, "Active/Active Topology"

  A more advanced topology allowing bi-direcitonal replication between two or more hosts.

  This topology can only be configured between MySQL hosts

- Section 2.7.3.3, "Fan-Out Topology"

  A more advanced Primary/Replica topology replicating from a single source host into multiple targets.

  Each target can be of a different type, and advanced filtering can elevate this topology into a highly advanced solution.

- Section 2.7.3.4, "Fan-In Topology"

  The reverse of Fan-Out, this topology allows multiple source hosts to be replicated into a single target.

  Advanced filtering within the replicator will allow flexibility to, for exmaple, remap schemas

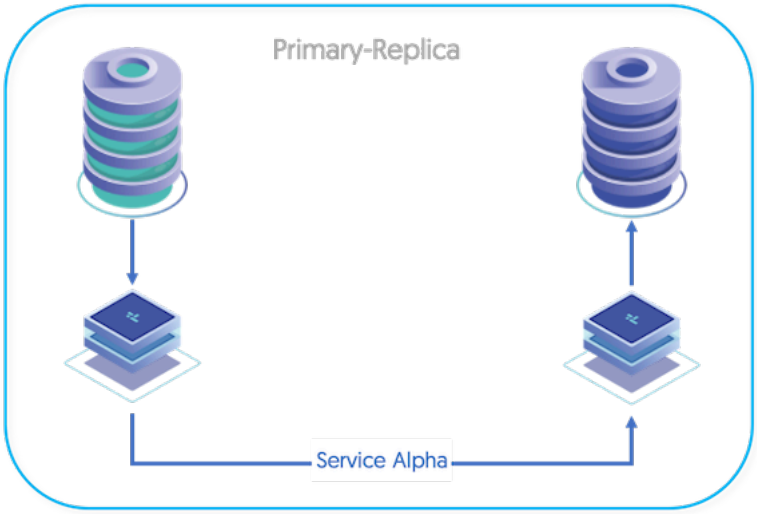
- Section 2.7.3.5, "Replicating in/out of an existing Tungsten Cluster"

Configuring the replicator as a Cluster-Extractor will allow you to leverage THL generated within an existing Tungsten Cluster to be replicated to a standalone target

### 2.7.3.1. Simple Primary/Replica Topology

Primary/Replica is the simplest and most straightforward of all replication scenarios, and also the basis of all other types of topology. The fundamental basis for the Primary/Replica topology is that changes in the Source are distributed and applied to the each of the configured Targets.

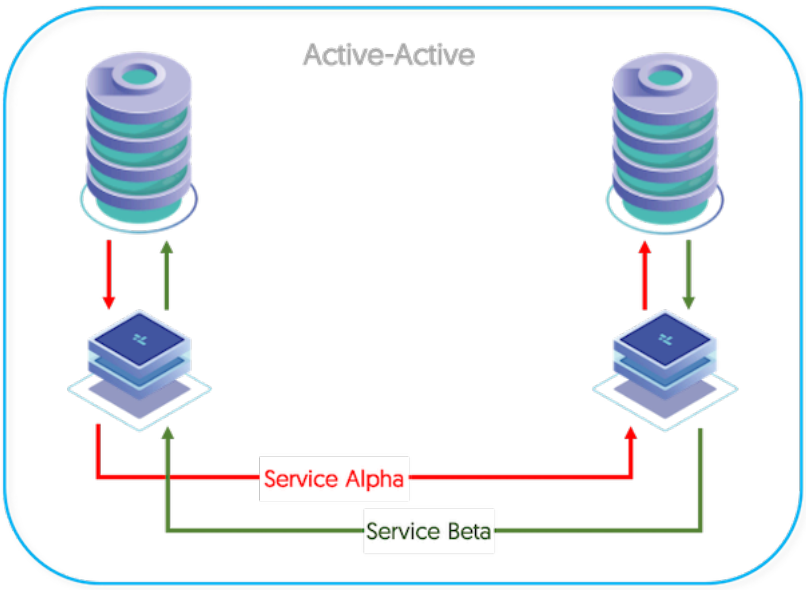
Figure 2.3. Topologies: Primary/Replica




### 2.7.3.2. Active/Active Topology

An active/active topology, relies on a number of individual services that are used to define a Primary/Replica topology between each group of hosts. In a three-node active/active setup, for example, three different services are created on each host, each service creates a Primary/Replica relationship between a primary host (itself) and the remote Targets. A change on any individual host will be replicated to the other databases in the topology creating the active/active configuration.

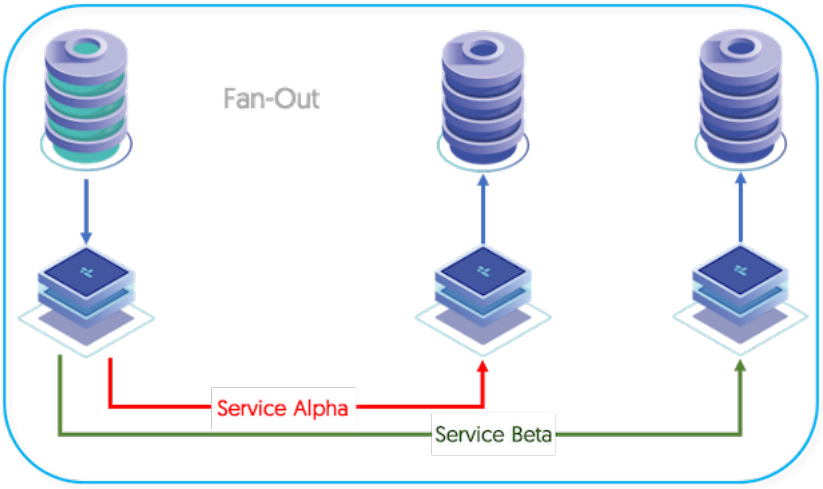
Figure 2.4. Topologies: Active/Active



### 2.7.3.3. Fan-Out Topology

The fan-out topology allows you to replicate from one single host out to two or more target hosts. Fan-out topologies are often in situations where you have different reporting requirements, for example, sales figures may need aggregating and reporting within a redshift environment but payroll information may need replicating to a MySQL environment for back office processing.

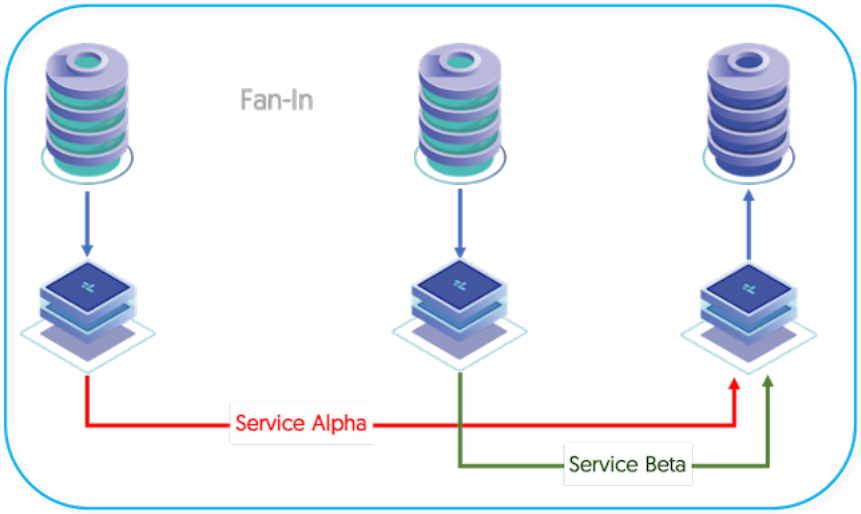
Figure 2.5. Topologies: Fan-Out




### 2.7.3.4. Fan-In Topology

The fan-in topology is the logical opposite of a Primary/Replica topology. In a fan-in topology, the data from two (or more) Sources is combined together on one Target. Fan-in topologies are often in situations where you have satellite databases, maybe for sales or retail operations, and need to combine that information together in a single database for processing.

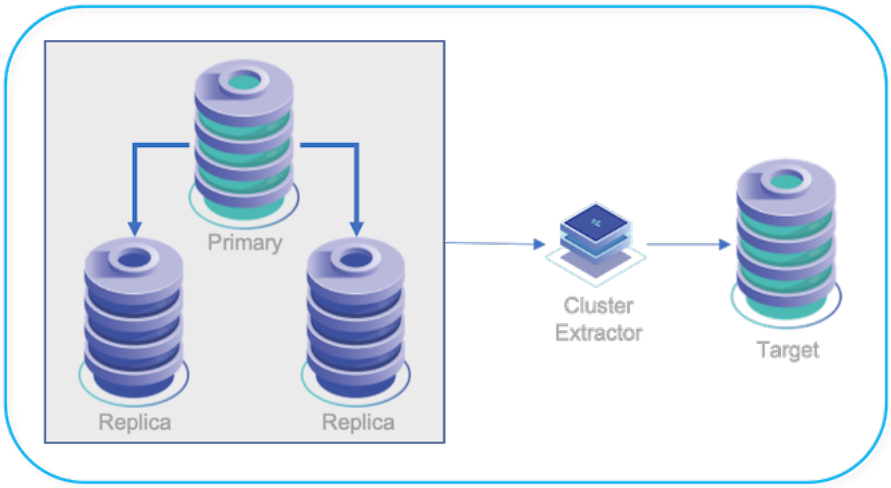
Figure 2.6. Topologies: Fan-In




### 2.7.3.5. Replicating in/out of an existing Tungsten Cluster

If you have an existing cluster and you want to replicate the data out to a separate standalone server using Tungsten Replicator then you can create a cluster alias, and use a Primary/Replica topology to replicate from the cluster. This allows for THL events from the cluster to be applied to a separate server for the purposes of backup or separate analysis.

Figure 2.7. Topologies: Cluster-Extractor




## 2.8. Understanding Heterogeneous Deployments

Heterogeneous deployments cover installations where data is being replicated between two different database solutions. These include, but are not limited to:

- MySQL (Incl. Cloud based solutions such as Amazon RDS, Aurora or Google Cloud), to...
  - ...Oracle
  - ...Hadoop
  - ...Amazon Redshift
  - ...Amazon RDS/Amazon Aurora
  - ...HP Vertica
  - ...Apache Kafka
  - ...MongoDB
  - ...Clickhouse
  - ...PostgreSQL

The following sections provide more detail and information on the setup and configuration of these different solutions.

### 2.8.1. How Heterogeneous Replication Works

Heterogeneous replication works slightly differently compared to the native MySQL to MySQL replication. This is because SQL statements, including both Data Manipulation Language (DML) and Data Definition Language (DDL) cannot be executed on a target system as they were extracted from the MySQL database. The SQL dialects are different, so that an SQL statement on MySQL is not the same as an SQL statement on Oracle, and differences in the dialects mean that either the statement would fail, or would perform an incorrect operation.

On targets that do not support SQL of any kind, such as MongoDB, replicating SQL statements would achieve nothing since they cannot be executed at all.

All heterogeneous replication deployments therefore use row-based replication. This extracts only the raw row data, not the statement information. Because it is only row-data, it can be easily re-assembled or constructed into another format, including statements in other SQL dialects, native appliers for alternative formats, such as JSON or BSON, or external CSV formats that enable the data to be loaded in bulk batches into a variety of different targets.

#### 2.8.1.1. JDBC Applier based Replication

Replication into targets where the JDBC Driver can be used, such as Oracle and Postgres, work as follows:

1. Data is extracted from the source MySQL database:

   - The MySQL server is configured to write transactions into the MySQL binary log using row-based logging. This generates information in the log in the form of the individual updated rows, rather than the statement that was used to perform the update. For example, instead of recording the statement:

     ```
     mysql> INSERT INTO MSG VALUES (1,'Hello World');
     ```

   - The information is stored as a row entry against the updated table:

     | | |
     |---|---|
     | 1 | Hello World |

   - The information is written into the THL as row-based events, with the event type (insert, update or delete) is appended to the metadata of the THL event.

   It is the raw row data that is stored in the THL. Because the row data, not the SQL statement, has been recorded, the differences in SQL dialects between does not need to be taken into account. In fact, Data Definition Language (DDL) and other SQL statements are deliberately ignored so that replication does not break.

2. The row-based transactions stored in the THL are transferred from the Extractor to the Applier.

3. On the Applier side, the row-based event data is wrapped into a suitable SQL statement for the target database environment. Because the raw row data is available, it can be constructed into any suitable statement appropriate for the target database.

### 2.8.1.2. Native Applier Replication (e.g. MongoDB)

For heterogeneous replication where data is written into a target database using a native applier, such as MongoDB, the row-based information is written into the database using the native API. With MongoDB, for example, data is reformatted into BSON and then applied into MongoDB using the native insert/update/delete API calls.

### 2.8.1.3. Batch Loading

For batch appliers, such as Hadoop, Vertica and Redshift, the row-data is converted into CSV files in batches. The format of the CSV file includes both the original row data for all the columns of each table, and metadata on each line that contain the unique *SEQNO* [549] and the operation type (insert, delete or update). A modified form of the CSV is used in some cases where the operation type is only an insert or delete, with updates being translated into a delete followed by an insert of the updated information.

These temporary CSV files are then loaded into the native environment as part of the replicator using a custom script that employs the specific tools of that database that support CSV imports. The raw CSV data is loaded into a staging table that contains the per-row metadata and the row data itself.

Depending on the batch environment, the loading of the data into the final destination tables is performed either within the same script, or by using a separate script. Both methods work in the same basic fashion; the base table is updated using the data from the staging table, with each row marked to be deleted, deleted, and the latest row (calculated from the highest *SEQNO* [549]) for each primary key) are then inserted

### 2.8.1.4. Schema Creation and Replication

Because heterogeneous replication does not replicate SQL statements, including DDL statements that would normally define and generate the table structures, a different method must be used.

Tungsten Replicator includes a tool called ddlscan which can read the schema definition from MySQL and translate that into the schema definition required on the target database. During the process, differences in supported sizes and datatypes are identified and either modified to a suitable value, or highlighted as a definition that must be changed in the generated DDL.

Once this modified form of the DDL has been completed, it can then be executed against the target database to generate the DDL required for Tungsten Replicator to apply data. The same basic method s used in batch loading environments where a staging table is required, with the additional staging columns added to the DDL automatically.

For MongoDB or Kafka, where no explicit DDL needs to be generated, the use of ddlscan is not required.

# Chapter 3. Deploying MySQL Extractors

The following sections outline the steps to configure the replicator for extraction. Each section covers the basic configuration to deploy an extractor in each of the deployment models (Onboard or Offboard) regardless of target database type.

To complete the deployment, after preparing the basic extractor configuration, follow the steps outlined in Chapter 4, *Deploying Appliers* appropriate to the target database type for your deployment.

## 3.1. MySQL Replication Pre-Requisites

Before installing Tungsten Replicator there are a number of steps that need to be completed to prepare the hosts.

First, ensure you have followed the general notes within Section B.3, "Host Configuration". For supported platforms and environments, see Section B.1, "Requirements".

If configuring extraction from MySQL instances hosted on your own hardware, or, for example, on EC2 instances, follow the MySQL specific pre-requisites within Section B.4, "MySQL Database Setup"

If configuring extraction from Amazon RDS or Amazon Aurora, also follow the pre-requisites within Section B.4, "MySQL Database Setup" however, paying specific attention to Section B.4.6, "MySQL Unprivileged Users"

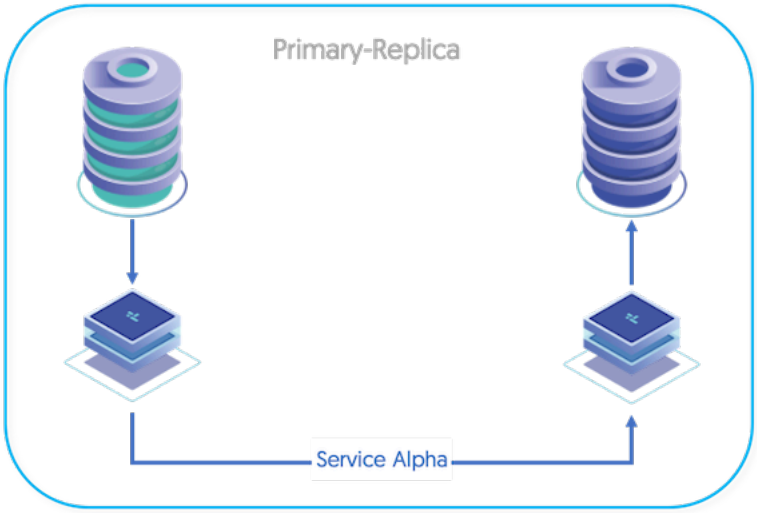
For more detail on changing parameters within Amazon AWS, see Section 3.3.1, "Changing Amazon RDS/Aurora Instance Configurations"

A pre-requisite checklist is available to download and can be used to ensure your environment is ready for installation. See Section B.5, "Pre-requisite Checklist"

## 3.2. Deploying a Primary/Replica Topology

Primary/Replica is the simplest and most straightforward of all replication scenarios, and also the basis of all other types of topology. The fundamental basis for the Primary/Replica topology is that changes in the Primary are distributed and applied to the each of the configured Replicas.

Figure 3.1. Topologies: Primary/Replica




This deployment style can be used against the following sources

- MySQL Community Edition
- MySQL Enterprise Edition
- Percona MySQL
- MariaDB
- Google Cloud MySQL

This deployment assumes full access to the host, including access to Binary Logs, therefore this deployment style is not suitable for RDS or Aurora extraction. For these sources, see Section 3.3, "Deploying an Extractor for Amazon Aurora"

tpm includes a specific topology structure for the basic Primary/Replica configuration, using the list of hosts and the Primary host definition to define the Primary/Replica relationship. Before starting the installation, the prerequisites must have been completed (see Appendix B, *Prerequisites*). To create a Primary/Replica using tpm:

There are two types of installation, either via a Staging Install, or via an ini file install.

To understand the differences between these two installation methods, see Section 9.1, "Comparing Staging and INI tpm Methods"

Regardless of which installation method you choose, the steps are the same, and are outlined below, using the appropriate example confguration based on your deployment style

### Note

If you plan to make full use of the REST API (which is enabled by default) you will need to also configure a username and password for API access. This must be done by specifying the following options in your configuration:

```
rest-api-admin-user=tungsten
rest-api-admin-pass=secret
```

- Install the Tungsten Replicator package (see Section 2.1.2, "Using the RPM package files"), or download the compressed tarball and unpack it, either on the source host, or on the staging host:

```
shell> cd /opt/continuent/software
shell> tar zxf tungsten-replicator-7.1.2-81.tar.gz
```

- Change to the Tungsten Replicator staging directory:

```
shell> cd tungsten-replicator-7.1.2-81
```

- Onboard Installation

  Configure the replicator for extraction from a locally installed and configured MySQL Installation (In this example, the service name is alpha)

  Click the link below to switch examples between Staging and INI methods

  Show Staging

  Show INI

```
shell> ./tools/tpm configure defaults \
    --reset \
    --install-directory=/opt/continuent \
    --user=tungsten \
    --profile-script=~/.bash_profile \
    --mysql-allow-intensive-checks=true \
    --rest-api-admin-user=apiuser \
    --rest-api-admin-pass=secret

shell> ./tools/tpm configure alpha \
    --master=localhost \
    --members=localhost \
    --enable-heterogeneous-service=true \
    --replication-port=3306 \
    --replication-user=tungsten_alpha \
    --replication-password=secret \
    --datasource-mysql-conf=/etc/my.cnf
```

```
shell> vi /etc/tungsten/tungsten.ini
```

```
[defaults]
install-directory=/opt/continuent
user=tungsten
profile-script=~/.bash_profile
mysql-allow-intensive-checks=true
rest-api-admin-user=apiuser
rest-api-admin-pass=secret

[alpha]
master=localhost
members=localhost
enable-heterogeneous-service=true
replication-port=3306
replication-user=tungsten_alpha
replication-password=secret
datasource-mysql-conf=/etc/my.cnf
```

  Configuration group `defaults`

The description of each of the options is shown below; click the icon to hide this detail:

Click the icon to show a detailed description of each argument.

- `--reset` [422]

  `reset` [422]

  For staging configurations, deletes all pre-existing configuration information between updating with the new configuration values.

- `--install-directory=/opt/continuent` [409]

  `install-directory=/opt/continuent` [409]

  Path to the directory where the active deployment will be installed. The configured directory will contain the software, THL and relay log information unless configured otherwise.

- `--user=tungsten` [429]

  `user=tungsten` [429]

  System User

- `--profile-script=~/.bash_profile` [417]

  `profile-script=~/.bash_profile` [417]

  Append commands to include env.sh in this profile script

- `--mysql-allow-intensive-checks=true` [413]

  `mysql-allow-intensive-checks=true` [413]

  For MySQL installation, enables detailed checks on the supported data types within the MySQL database to confirm compatibility. This includes checking each table definition individually for any unsupported data types.

- `--rest-api-admin-user=apiuser` [422]

  `rest-api-admin-user=apiuser` [422]
  Optional: Must be specified along with rest-api-admin-pass if you wish to access the full API features and use the Dashboard GUI for cluster installations.

- `--rest-api-admin-pass=secret` [422]

  `rest-api-admin-pass=secret` [422]
  Optional: Must be specified along with rest-api-admin-user if you wish to access the full API features.

Configuration group **`alpha`**

The description of each of the options is shown below; click the icon to hide this detail:

Click the icon to show a detailed description of each argument.

- `--master=localhost` [412]

  `master=localhost` [412]

  The hostname of the primary (extractor) within the current service.

- `--members=localhost` [413]

  `members=localhost` [413]

  Hostnames for the dataservice members

- `--enable-heterogeneous-service=true` [406]

  `enable-heterogeneous-service=true` [406]

  - On a Primary
    - `--mysql-use-bytes-for-string` [414] is set to false.

    - colnames filter is enabled (in the binlog-to-q stage to add column names to the THL information.
    - pkey filter is enabled (in the binlog-to-q and q-to-dbms stage), with the *addPkeyToInserts* and *addColumnsToDeletes* filter options set to false.
    - enumtostring filter is enabled (in the q-to-thl stage), to translate ENUM values to their string equivalents.
    - settostring filter is enabled (in the q-to-thl stage), to translate SET values to their string equivalents.
  - On a Replica
    - --mysql-use-bytes-for-string [414] is set to true.
    - pkey filter is enabled (q-to-dbms stage).
- --replication-port=3306 [421]

  replication-port=3306 [421]

  The network port used to connect to the database server. The default port used depends on the database being configured.
- --replication-user=tungsten_alpha [421]

  replication-user=tungsten_alpha [421]

  For databases that required authentication, the username to use when connecting to the database using the corresponding connection method (native, JDBC, etc.).
- --replication-password=secret [420]

  replication-password=secret [420]

  The password to be used when connecting to the database using the corresponding --replication-user [421].
- --datasource-mysql-conf=/etc/my.cnf [401]

  datasource-mysql-conf=/etc/my.cnf [401]

  MySQL config file

In the above example, --datasource-mysql-conf [401], is optional and can be used if the MySQL configuration file cannot be located by tpm, or is in a non-default location

- Offboard Installation

  Configure the replicator for extraction from a remotely installed and configured MySQL Installation (In this example, the service name is alpha)

  In the example below, the server offboardhost is the host that the Replicator is installed upon, and the server dbhost is the database host to apply the events to.

  Click the link below to switch examples between Staging and INI methods

  Show Staging

  Show INI

```
shell> ./tools/tpm configure defaults \
    --reset \
    --install-directory=/opt/continuent \
    --user=tungsten \
    --profile-script=~/.bash_profile \
    --mysql-allow-intensive-checks=true \
    --skip-validation-check=MySQLAvailableCheck \
    --skip-validation-check=MySQLConfFile \
    --skip-validation-check=RowBasedBinaryLoggingCheck \
    --rest-api-admin-user=apiuser \
    --rest-api-admin-pass=secret

shell> ./tools/tpm configure alpha \
    --master=offboardhost \
    --members=offboardhost \
    --enable-heterogeneous-service=true \
    --privileged-master=true \
    --replication-host=dbhost \
    --replication-port=3306 \
```

```
    --replication-user=tungsten_alpha \
    --replication-password=secret
```

```
shell> vi /etc/tungsten/tungsten.ini
```

```
[defaults]
install-directory=/opt/continuent
user=tungsten
profile-script=~/.bash_profile
mysql-allow-intensive-checks=true
skip-validation-check=MySQLAvailableCheck
skip-validation-check=MySQLConfFile
skip-validation-check=RowBasedBinaryLoggingCheck
rest-api-admin-user=apiuser
rest-api-admin-pass=secret

[alpha]
master=offboardhost
members=offboardhost
enable-heterogeneous-service=true
privileged-master=true
replication-host=dbhost
replication-port=3306
replication-user=tungsten_alpha
replication-password=secret
```

Configuration group **defaults**

The description of each of the options is shown below; click the icon to hide this detail:

Click the icon to show a detailed description of each argument.

- `--reset` [422]

  `reset` [422]

  For staging configurations, deletes all pre-existing configuration information between updating with the new configuration values.

- `--install-directory=/opt/continuent` [409]

  `install-directory=/opt/continuent` [409]

  Path to the directory where the active deployment will be installed. The configured directory will contain the software, THL and relay log information unless configured otherwise.

- `--user=tungsten` [429]

  `user=tungsten` [429]

  System User

- `--profile-script=~/.bash_profile` [417]

  `profile-script=~/.bash_profile` [417]

  Append commands to include env.sh in this profile script

- `--mysql-allow-intensive-checks=true` [413]

  `mysql-allow-intensive-checks=true` [413]

  For MySQL installation, enables detailed checks on the supported data types within the MySQL database to confirm compatibility. This includes checking each table definition individually for any unsupported data types.

- `--skip-validation-check=MySQLAvailableCheck` [369]

  `skip-validation-check=MySQLAvailableCheck` [369]

  The `--skip-validation-check` [369] disables a given validation check. If any validation check fails, the installation, validation or configuration will automatically stop.

  **Warning**

  Using this option enables you to bypass the specified check, although skipping a check may lead to an invalid or non-working configuration.

You can identify a given check if an error or warning has been raised during configuration. For example, the default table type check:

```
...
ERROR >> centos >> The datasource root@centos:3306 (WITH PASSWORD) »
 uses MyISAM as the default storage engine (MySQLDefaultTableTypeCheck)
...
```

The check in this case is `MySQLDefaultTableTypeCheck` [380], and could be ignored using `--skip-validation-check=MySQLDefaultTableTypeCheck` [369].

Setting both `--skip-validation-check` [369] and `--enable-validation-check` [367] is equivalent to explicitly disabling the specified check.

- `--skip-validation-check=MySQLConfFile` [369]

  `skip-validation-check=MySQLConfFile` [369]

  The `--skip-validation-check` [369] disables a given validation check. If any validation check fails, the installation, validation or configuration will automatically stop.

  > **Warning**
  >
  > Using this option enables you to bypass the specified check, although skipping a check may lead to an invalid or non-working configuration.

  You can identify a given check if an error or warning has been raised during configuration. For example, the default table type check:

  ```
  ...
  ERROR >> centos >> The datasource root@centos:3306 (WITH PASSWORD) »
   uses MyISAM as the default storage engine (MySQLDefaultTableTypeCheck)
  ...
  ```

  The check in this case is `MySQLDefaultTableTypeCheck` [380], and could be ignored using `--skip-validation-check=MySQLDefaultTableTypeCheck` [369].

  Setting both `--skip-validation-check` [369] and `--enable-validation-check` [367] is equivalent to explicitly disabling the specified check.

- `--skip-validation-check=RowBasedBinaryLoggingCheck` [369]

  `skip-validation-check=RowBasedBinaryLoggingCheck` [369]

  The `--skip-validation-check` [369] disables a given validation check. If any validation check fails, the installation, validation or configuration will automatically stop.

  > **Warning**
  >
  > Using this option enables you to bypass the specified check, although skipping a check may lead to an invalid or non-working configuration.

  You can identify a given check if an error or warning has been raised during configuration. For example, the default table type check:

  ```
  ...
  ERROR >> centos >> The datasource root@centos:3306 (WITH PASSWORD) »
   uses MyISAM as the default storage engine (MySQLDefaultTableTypeCheck)
  ...
  ```

  The check in this case is `MySQLDefaultTableTypeCheck` [380], and could be ignored using `--skip-validation-check=MySQLDefaultTableTypeCheck` [369].

  Setting both `--skip-validation-check` [369] and `--enable-validation-check` [367] is equivalent to explicitly disabling the specified check.

- `--rest-api-admin-user=apiuser` [422]

  `rest-api-admin-user=apiuser` [422]
  Optional: Must be specified along with rest-api-admin-pass if you wish to access the full API features and use the Dashboard GUI for cluster installations.

- `--rest-api-admin-pass=secret` [422]

  `rest-api-admin-pass=secret` [422]
  Optional: Must be specified along with rest-api-admin-user if you wish to access the full API features.

Configuration group `alpha`

The description of each of the options is shown below; click the icon to hide this detail:

Click the icon to show a detailed description of each argument.

- `--master=offboardhost` [412]

  `master=offboardhost` [412]

  The hostname of the primary (extractor) within the current service.

- `--members=offboardhost` [413]

  `members=offboardhost` [413]

  Hostnames for the dataservice members

- `--enable-heterogeneous-service=true` [406]

  `enable-heterogeneous-service=true` [406]

  - On a Primary
    - `--mysql-use-bytes-for-string` [414] is set to false.
    - `colnames` filter is enabled (in the `binlog-to-q` stage to add column names to the THL information.
    - `pkey` filter is enabled (in the `binlog-to-q` and `q-to-dbms` stage), with the *`addPkeyToInserts`* and *`addColumnsToDeletes`* filter options set to false.
    - `enumtostring` filter is enabled (in the `q-to-thl` stage), to translate `ENUM` values to their string equivalents.
    - `settostring` filter is enabled (in the `q-to-thl` stage), to translate `SET` values to their string equivalents.
  - On a Replica
    - `--mysql-use-bytes-for-string` [414] is set to true.
    - `pkey` filter is enabled (`q-to-dbms` stage).

- `--privileged-master=true` [417]

  `privileged-master=true` [417]

  Does the login for the Primary database service have superuser privileges

- `--replication-host=dbhost` [420]

  `replication-host=dbhost` [420]

  Hostname of the datasource where the database is located. If the specified hostname matches the current host or member name, the database is assumed to be local. If the hostnames do not match, extraction is assumed to be via remote access. For MySQL hosts, this configures a remote replication Replica (relay) connection.

- `--replication-port=3306` [421]

  `replication-port=3306` [421]

  The network port used to connect to the database server. The default port used depends on the database being configured.

- `--replication-user=tungsten_alpha` [421]

  `replication-user=tungsten_alpha` [421]

  For databases that required authentication, the username to use when connecting to the database using the corresponding connection method (native, JDBC, etc.).

- `--replication-password=secret` [420]

  `replication-password=secret` [420]

  The password to be used when connecting to the database using the corresponding `--replication-user` [421].



- Once the prerequisites and configuring of the installation has been completed, the software can be installed:

```
shell> ./tools/tpm install
```

In both of the above examples, `enable-heterogenous-service`, is only required if the target applier is NOT a MySQL database

If the installation process fails, check the output of the `/tmp/tungsten-configure.log` file for more information about the root cause.

Once the installation has been completed, you can now proceed to configure the Applier service following the relevant step within Chapter 4, *Deploying Appliers*.

Following installation of the applier, the services can be started. For information on starting and stopping Tungsten Cluster see Section 2.4, "Starting and Stopping Tungsten Replicator"; configuring init scripts to startup and shutdown when the system boots and shuts down, see Section 2.5, "Configuring Startup on Boot".

For information on checking the running service, see Section 3.2.1, "Monitoring the MySQL Extractor".

## 3.2.1. Monitoring the MySQL Extractor

Once the service has been started, a quick view of the service status can be determined using trepctl:

```
shell> trepctl services
Processing services command...
NAME              VALUE
----              -----
appliedLastSeqno: 3593
appliedLatency  : 1.074
role            : master
serviceName     : alpha
serviceType     : local
started         : true
state           : ONLINE
Finished services command...
```

The key fields are:

- *`appliedLastSeqno`* and *`appliedLatency`* indicate the global transaction ID and latency of the host. These are important when monitoring the status of the cluster to determine how up to date a host is and whether a specific transaction has been applied.
- *`role`* indicates the current role of the host within the scope of this dataservice.
- *`state`* shows the current status of the host within the scope of this dataservice.

More detailed status information can also be obtained. On the Extractor:

```
shell> trepctl status
Processing status command...
NAME                     VALUE
----                     -----
appliedLastEventId     : mysql-bin.000009:0000000000001033;0
appliedLastSeqno       : 3593
appliedLatency         : 1.074
channels               : 1
clusterName            : default
currentEventId         : mysql-bin.000009:0000000000001033
currentTimeMillis      : 1373615598598
dataServerHost         : host1
extensions             :
latestEpochNumber      : 3589
masterConnectUri       :
masterListenUri        : thl://host1:2112/
maximumStoredSeqNo     : 3593
minimumStoredSeqNo     : 0
offlineRequests        : NONE
pendingError           : NONE
pendingErrorCode       : NONE
pendingErrorEventId    : NONE
pendingErrorSeqno      : -1
pendingExceptionMessage: NONE
pipelineSource         : jdbc:mysql:thin://host1:3306/
relativeLatency        : 604904.598
resourcePrecedence     : 99
rmiPort                : 10000
role                   : master
seqnoType              : java.lang.Long
serviceName            : alpha
serviceType            : local
simpleServiceName      : alpha
siteName               : default
sourceId               : host1
state                  : ONLINE
timeInStateSeconds     : 604903.621
```

```
transitioningTo        :
uptimeSeconds          : 1202137.328
version                : Tungsten Replicator 7.1.2 build 81
Finished status command...
```

For more information on using trepctl, see Section 8.20, "The trepctl Command".

Definitions of the individual field descriptions in the above example output can be found in Section E.2, "Generated Field Reference".

For more information on management and operational detailed for managing your replicator installation, see Chapter 7, *Operations Guide*.

# 3.3. Deploying an Extractor for Amazon Aurora

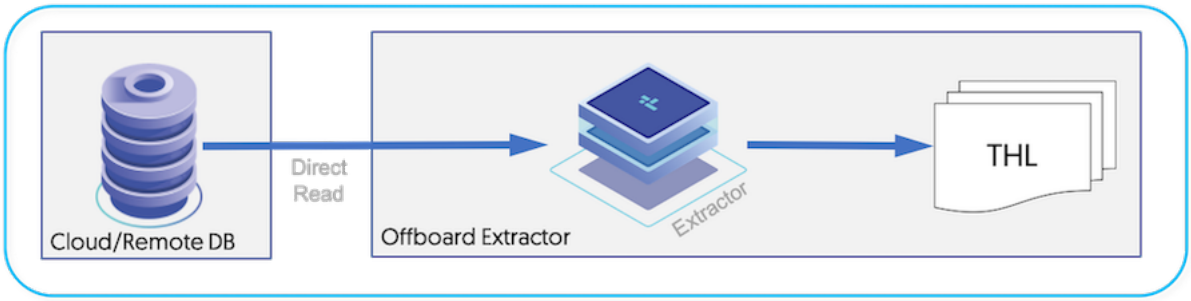
Replicating from Amazon Aurora, operates by directly accessing the binary log provided by Aurora and enables you to take advantage of the Amazon Web, either replicating from the remote Aurora instance, or to a standard EC2 instance within AWS. The complexity with Aurora is that there is no access to the host that is running the instance, or the MySQL binary logs.

To use this service, two aspects of the Tungsten Replicator are required, direct mode and unprivileged user support. Direct mode reads the MySQL binary log over the network, rather than accessing the binlog on the filesystem. The unprivileged mode enables the user to access and update information within Aurora without requiring SUPER privileges, which are unavailable within an Aurora instance. For more information, see Section B.4.6, "MySQL Unprivileged Users".

The deployment requires a host for the extractor installation, this can be an EC2 instance within your AWS environment, or it could be a remote host in your own environment.

This deployment follows a similar model to an Offboard Installation

Figure 3.2. Topologies: Aurora Extraction




Before starting the installation, the prerequisites must have been completed (see Appendix B, *Prerequisites*) on both the Host designated for the installation of the extractor, and within the source database instance.

There are two types of installation, either via a Staging Install, or via an ini file install.

To understand the differences between these two installation methods, see Section 9.1, "Comparing Staging and INI tpm Methods"

Regardless of which installation method you choose, the steps are the same, and are outlined below.

- Install the Tungsten Replicator package (see Section 2.1.2, "Using the RPM package files"), or download the compressed tarball and unpack it, either on the source host, or on the staging host:

  ```
  shell> cd /opt/continuent/software
  shell> tar zxf tungsten-replicator-7.1.2-81.tar.gz
  ```

- Change to the Tungsten Replicator staging directory:

  ```
  shell> cd tungsten-replicator-7.1.2-81
  ```

- Configure the replicator for extraction (In this example, the service name is alpha)

  Click the link below to switch examples between Staging and INI methods

  Show Staging

  Show INI

  ```
  shell> ./tools/tpm configure defaults \
      --reset \
      --install-directory=/opt/continuent \
      --user=tungsten \
      --profile-script=~/.bash_profile \
      --mysql-allow-intensive-checks=true \
  ```

```
    --skip-validation-check=InstallerMasterSlaveCheck \
    --skip-validation-check=MySQLPermissionsCheck \
    --skip-validation-check=MySQLBinaryLogsEnabledCheck \
    --skip-validation-check=MySQLMyISAMCheck \
    --skip-validation-check=RowBasedBinaryLoggingCheck \
    --rest-api-admin-user=apiuser \
    --rest-api-admin-pass=secret

shell> ./tools/tpm configure alpha \
    --master=localhost \
    --members=localhost \
    --enable-heterogeneous-service=true \
    --privileged-master=false \
    --replication-host=rds.endpoint.url \
    --replication-port=3306 \
    --replication-user=tungsten_alpha \
    --replication-password=secret \
    --datasource-mysql-conf=/dev/null \
    --svc-extractor-filters=dropcatalogdata \
    --property=replicator.service.comments=true
```

```
shell> vi /etc/tungsten/tungsten.ini
```

```
[defaults]
install-directory=/opt/continuent
user=tungsten
profile-script=~/.bash_profile
mysql-allow-intensive-checks=true
skip-validation-check=InstallerMasterSlaveCheck
skip-validation-check=MySQLPermissionsCheck
skip-validation-check=MySQLBinaryLogsEnabledCheck
skip-validation-check=MySQLMyISAMCheck
skip-validation-check=RowBasedBinaryLoggingCheck
rest-api-admin-user=apiuser
rest-api-admin-pass=secret

[alpha]
master=localhost
members=localhost
enable-heterogeneous-service=true
privileged-master=false
replication-host=rds.endpoint.url
replication-port=3306
replication-user=tungsten_alpha
replication-password=secret
datasource-mysql-conf=/dev/null
svc-extractor-filters=dropcatalogdata
property=replicator.service.comments=true
```

Configuration group **`defaults`**

The description of each of the options is shown below; click the icon to show this detail:

Click the icon to show a detailed description of each argument.

- `--reset` [422]

  `reset` [422]

  For staging configurations, deletes all pre-existing configuration information between updating with the new configuration values.

- `--install-directory=/opt/continuent` [409]

  `install-directory=/opt/continuent` [409]

  Path to the directory where the active deployment will be installed. The configured directory will contain the software, THL and relay log information unless configured otherwise.

- `--user=tungsten` [429]

  `user=tungsten` [429]

  System User

- `--profile-script=~/.bash_profile` [417]

  `profile-script=~/.bash_profile` [417]

  Append commands to include env.sh in this profile script

- `--mysql-allow-intensive-checks=true` [413]

  `mysql-allow-intensive-checks=true` [413]

  For MySQL installation, enables detailed checks on the supported data types within the MySQL database to confirm compatibility. This includes checking each table definition individually for any unsupported data types.

- `--skip-validation-check=InstallerMasterSlaveCheck` [369]

  `skip-validation-check=InstallerMasterSlaveCheck` [369]

  The `--skip-validation-check` [369] disables a given validation check. If any validation check fails, the installation, validation or configuration will automatically stop.

  > **Warning**
  >
  > Using this option enables you to bypass the specified check, although skipping a check may lead to an invalid or non-working configuration.

  You can identify a given check if an error or warning has been raised during configuration. For example, the default table type check:

  ```
  ...
  ERROR >> centos >> The datasource root@centos:3306 (WITH PASSWORD) »
   uses MyISAM as the default storage engine (MySQLDefaultTableTypeCheck)
  ...
  ```

  The check in this case is `MySQLDefaultTableTypeCheck` [380], and could be ignored using `--skip-validation-check=MySQLDefaultTableTypeCheck` [369].

  Setting both `--skip-validation-check` [369] and `--enable-validation-check` [367] is equivalent to explicitly disabling the specified check.

- `--skip-validation-check=MySQLPermissionsCheck` [369]

  `skip-validation-check=MySQLPermissionsCheck` [369]

  The `--skip-validation-check` [369] disables a given validation check. If any validation check fails, the installation, validation or configuration will automatically stop.

  > **Warning**
  >
  > Using this option enables you to bypass the specified check, although skipping a check may lead to an invalid or non-working configuration.

  You can identify a given check if an error or warning has been raised during configuration. For example, the default table type check:

  ```
  ...
  ERROR >> centos >> The datasource root@centos:3306 (WITH PASSWORD) »
   uses MyISAM as the default storage engine (MySQLDefaultTableTypeCheck)
  ...
  ```

  The check in this case is `MySQLDefaultTableTypeCheck` [380], and could be ignored using `--skip-validation-check=MySQLDefaultTableTypeCheck` [369].

  Setting both `--skip-validation-check` [369] and `--enable-validation-check` [367] is equivalent to explicitly disabling the specified check.

- `--skip-validation-check=MySQLBinaryLogsEnabledCheck` [369]

  `skip-validation-check=MySQLBinaryLogsEnabledCheck` [369]

  The `--skip-validation-check` [369] disables a given validation check. If any validation check fails, the installation, validation or configuration will automatically stop.

  > **Warning**
  >
  > Using this option enables you to bypass the specified check, although skipping a check may lead to an invalid or non-working configuration.

  You can identify a given check if an error or warning has been raised during configuration. For example, the default table type check:

  ```
  ...
  ERROR >> centos >> The datasource root@centos:3306 (WITH PASSWORD) »
   uses MyISAM as the default storage engine (MySQLDefaultTableTypeCheck)
  ```

```
...
```

The check in this case is `MySQLDefaultTableTypeCheck` [380], and could be ignored using `--skip-validation-check=MySQLDefaultTableTypeCheck` [369].

Setting both `--skip-validation-check` [369] and `--enable-validation-check` [367] is equivalent to explicitly disabling the specified check.

- `--skip-validation-check=MySQLMyISAMCheck` [369]

  `skip-validation-check=MySQLMyISAMCheck` [369]

  The `--skip-validation-check` [369] disables a given validation check. If any validation check fails, the installation, validation or configuration will automatically stop.

  > **Warning**
  >
  > Using this option enables you to bypass the specified check, although skipping a check may lead to an invalid or non-working configuration.

  You can identify a given check if an error or warning has been raised during configuration. For example, the default table type check:

```
...
ERROR >> centos >> The datasource root@centos:3306 (WITH PASSWORD) »
 uses MyISAM as the default storage engine (MySQLDefaultTableTypeCheck)
...
```

  The check in this case is `MySQLDefaultTableTypeCheck` [380], and could be ignored using `--skip-validation-check=MySQLDefaultTableTypeCheck` [369].

  Setting both `--skip-validation-check` [369] and `--enable-validation-check` [367] is equivalent to explicitly disabling the specified check.

- `--skip-validation-check=RowBasedBinaryLoggingCheck` [369]

  `skip-validation-check=RowBasedBinaryLoggingCheck` [369]

  The `--skip-validation-check` [369] disables a given validation check. If any validation check fails, the installation, validation or configuration will automatically stop.

  > **Warning**
  >
  > Using this option enables you to bypass the specified check, although skipping a check may lead to an invalid or non-working configuration.

  You can identify a given check if an error or warning has been raised during configuration. For example, the default table type check:

```
...
ERROR >> centos >> The datasource root@centos:3306 (WITH PASSWORD) »
 uses MyISAM as the default storage engine (MySQLDefaultTableTypeCheck)
...
```

  The check in this case is `MySQLDefaultTableTypeCheck` [380], and could be ignored using `--skip-validation-check=MySQLDefaultTableTypeCheck` [369].

  Setting both `--skip-validation-check` [369] and `--enable-validation-check` [367] is equivalent to explicitly disabling the specified check.

- `--rest-api-admin-user=apiuser` [422]

  `rest-api-admin-user=apiuser` [422]
  Optional: Must be specified along with rest-api-admin-pass if you wish to access the full API features and use the Dashboard GUI for cluster installations.

- `--rest-api-admin-pass=secret` [422]

  `rest-api-admin-pass=secret` [422]
  Optional: Must be specified along with rest-api-admin-user if you wish to access the full API features.

Configuration group **`alpha`**

The description of each of the options is shown below; click the icon to hide this detail:

Click the icon to show a detailed description of each argument.



- `--master=localhost` [412]

`master=localhost` [412]

The hostname of the primary (extractor) within the current service.

- `--members=localhost` [413]

  `members=localhost` [413]

  Hostnames for the dataservice members

- `--enable-heterogeneous-service=true` [406]

  `enable-heterogeneous-service=true` [406]

  - On a Primary
    - `--mysql-use-bytes-for-string` [414] is set to false.
    - `colnames` filter is enabled (in the `binlog-to-q` stage to add column names to the THL information.
    - `pkey` filter is enabled (in the `binlog-to-q` and `q-to-dbms` stage), with the *`addPkeyToInserts`* and *`addColumnsToDeletes`* filter options set to false.
    - `enumtostring` filter is enabled (in the `q-to-thl` stage), to translate `ENUM` values to their string equivalents.
    - `settostring` filter is enabled (in the `q-to-thl` stage), to translate `SET` values to their string equivalents.
  - On a Replica
    - `--mysql-use-bytes-for-string` [414] is set to true.
    - `pkey` filter is enabled (`q-to-dbms` stage).

- `--privileged-master=false` [417]

  `privileged-master=false` [417]

  Does the login for the Primary database service have superuser privileges

- `--replication-host=rds.endpoint.url` [420]

  `replication-host=rds.endpoint.url` [420]

  Hostname of the datasource where the database is located. If the specified hostname matches the current host or member name, the database is assumed to be local. If the hostnames do not match, extraction is assumed to be via remote access. For MySQL hosts, this configures a remote replication Replica (relay) connection.

- `--replication-port=3306` [421]

  `replication-port=3306` [421]

  The network port used to connect to the database server. The default port used depends on the database being configured.

- `--replication-user=tungsten_alpha` [421]

  `replication-user=tungsten_alpha` [421]

  For databases that required authentication, the username to use when connecting to the database using the corresponding connection method (native, JDBC, etc.).

- `--replication-password=secret` [420]

  `replication-password=secret` [420]

  The password to be used when connecting to the database using the corresponding `--replication-user` [421].

- `--datasource-mysql-conf=/dev/null` [401]

  `datasource-mysql-conf=/dev/null` [401]

  MySQL config file



- `--svc-extractor-filters=dropcatalogdata` [425]

svc-extractor-filters=dropcatalogdata [425]

Replication service extractor filters

- --property=replicator.service.comments=true [368]

  property=replicator.service.comments=true [368]

  The --property [368] option enables you to explicitly set property values in the target files. A number of different models are supported:

  - key=value

    Set the property defined by key to the specified value without evaluating any template values or other rules.

  - key+=value

    Add the value to the property defined by key. Template values and other options append their settings to the end of the specified property.

  - key~=/match/replace/

    Evaluate any template values and other settings, and then perform the specified Ruby regex operation to the property defined by key. For example --property=replicator.key~=/(.*)/somevalue,\1/ will prepend somevalue before the template value for replicator.key.

- Once the prerequisites and configuring of the installation has been completed, the software can be installed:

```
shell> ./tools/tpm install
```

In the above examples,

- enable-heterogenous-service, is only required if the target applier is NOT a MySQL database
- datasource-mysql-conf [401], needs to be set as shown as we do not have access to the my.cnf file

If the installation process fails, check the output of the /tmp/tungsten-configure.log file for more information about the root cause.

Once the installation has been completed, you can now proceed to configure the Applier service following the relevant step within Chapter 4, *Deploying Appliers*.

Following installation of the applier, the services can be started. For information on starting and stopping Tungsten Cluster see Section 2.4, "Starting and Stopping Tungsten Replicator"; configuring init scripts to startup and shutdown when the system boots and shuts down, see Section 2.5, "Configuring Startup on Boot".

Monitoring the extractor is the same as an extractor from MySQL, for information, see Section 3.2.1, "Monitoring the MySQL Extractor".

## 3.3.1. Changing Amazon RDS/Aurora Instance Configurations

The configuration of RDS and Aurora instances can be modified to change the parameters for MySQL instances, the Amazon equivalent of modifying the my.cnf file.

### 3.3.1.1. Changing Amazon RDS using command line functions

These steps can be used for changing the configuration for RDS Instances only. See Section 3.3.1.2, "Changing Amazon Aurora Parameters using AWS Console" for steps to change Aurora parameters

The parameters can be set internally by connecting to the instance and using the configuration function within the instance. For example:

```
mysql> call mysql.rds_set_configuration('binlog retention hours', 48);
```

An RDS command-line interface is available which enables modifying these parameters. To enable the command-line interface:

```
shell> wget http://s3.amazonaws.com/rds-downloads/RDSCli.zip
shell> unzip RDSCli.zip
shell> export AWS_RDS_HOME=/home/tungsten/RDSCli-1.13.002
shell> export PATH=$PATH:$AWS_RDS_HOME/bin
```

The current RDS instances can be listed by using rds-describe-db-instances:

```
shell> rds-describe-db-instances --region=us-east-1
```

To change parameters, a new parameter group must be created, and then applied to a running instance or instances before restarting the instance:

1. Create a new custom parameter group:

```
shell> rds-create-db-parameter-group repgroup -d 'Parameter group for DB Replicas' -f mysql5.1
```

   Where *repgroup* is the replicator group name.

2. Set the new parameter value:

```
shell> rds-modify-db-parameter-group repgroup --parameters \
"name=max_allowed_packet,value=67108864, method=immediate"
```

3. Apply the parameter group to your instance:

```
shell> rds-modify-db-instance instancename --db-parameter-group-name=repgroup
```

   Where *instancename* is the name given to your instance.

4. Restart the instance:

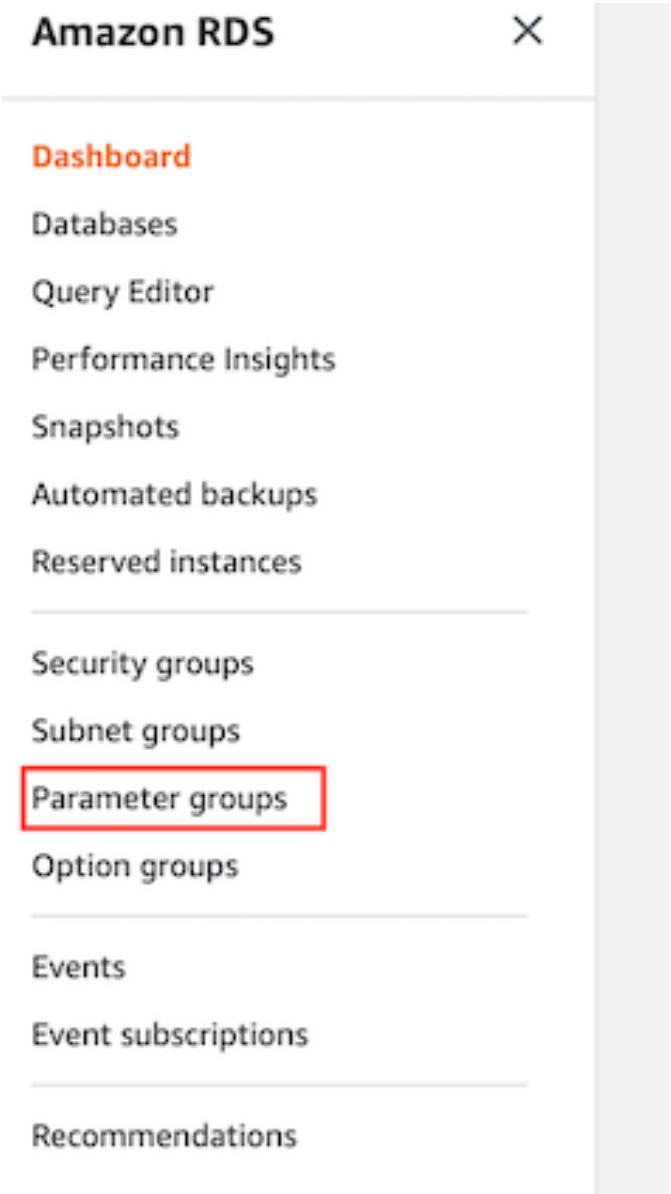
```
shell> rds-reboot-db-instance instancename
```

### 3.3.1.2. Changing Amazon Aurora Parameters using AWS Console

To change the parameters for Aurora Instances, you can follow the following guidelines using the AWS Console

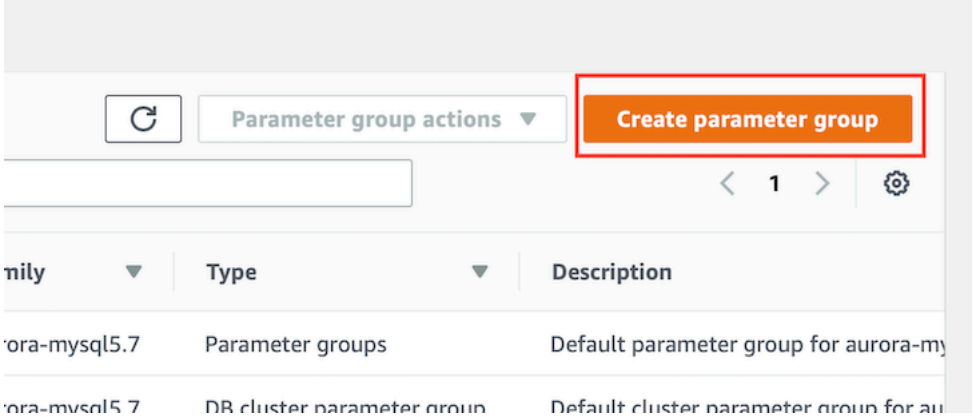
1. Login to the AWS Console using your account credentials and navigate to the RDS Dashboard. From here, select "Parameter Groups" from the left hand list

Figure 3.3. Fig 1. AWS Config




2. Select the "Create Parameter Group" Button to the top right

Figure 3.4. Fig 2. AWS Config




3. This dialog will now allow you to create a new parameter group using an existing one as a template. Select the appropriate template to use and complete the rest of the details. You need to create a DB Paramater group and a DB Cluster Parameter Group

Figure 3.5. Fig 3. AWS Config

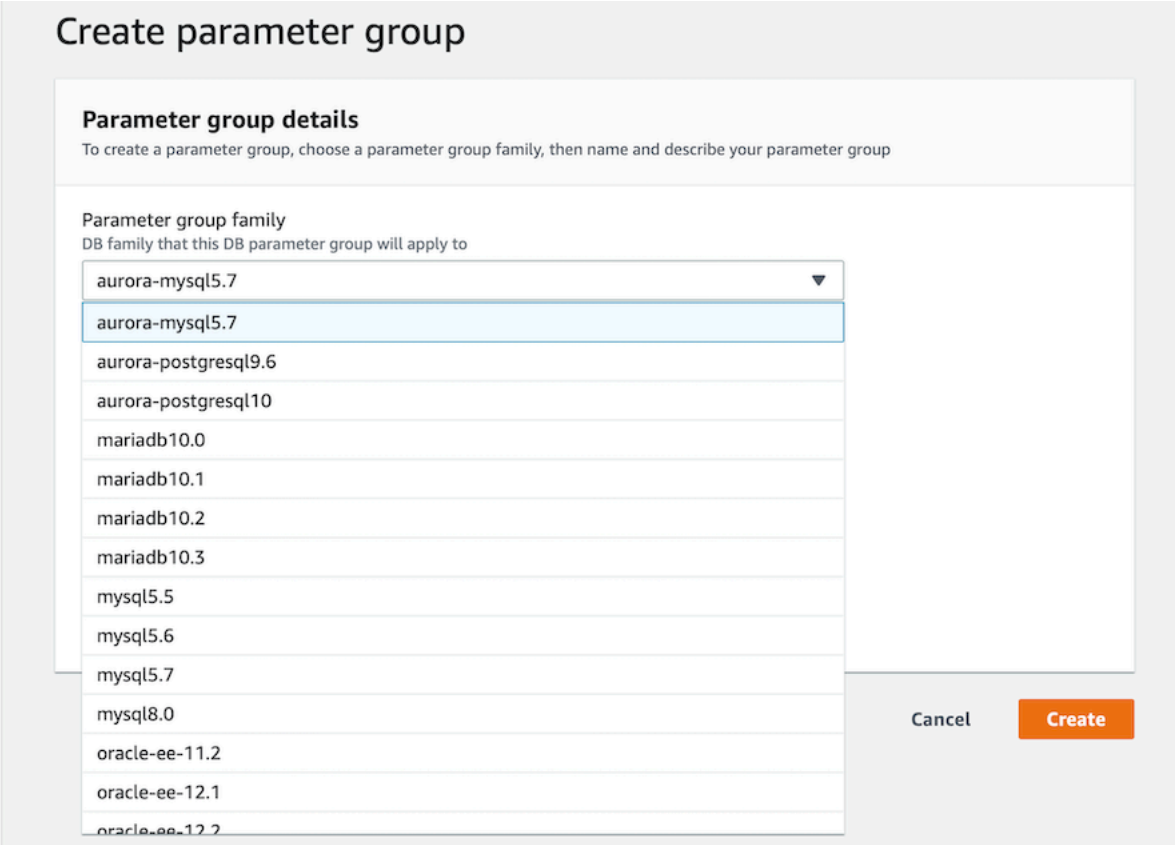


Figure 3.6. Fig 4. AWS Config

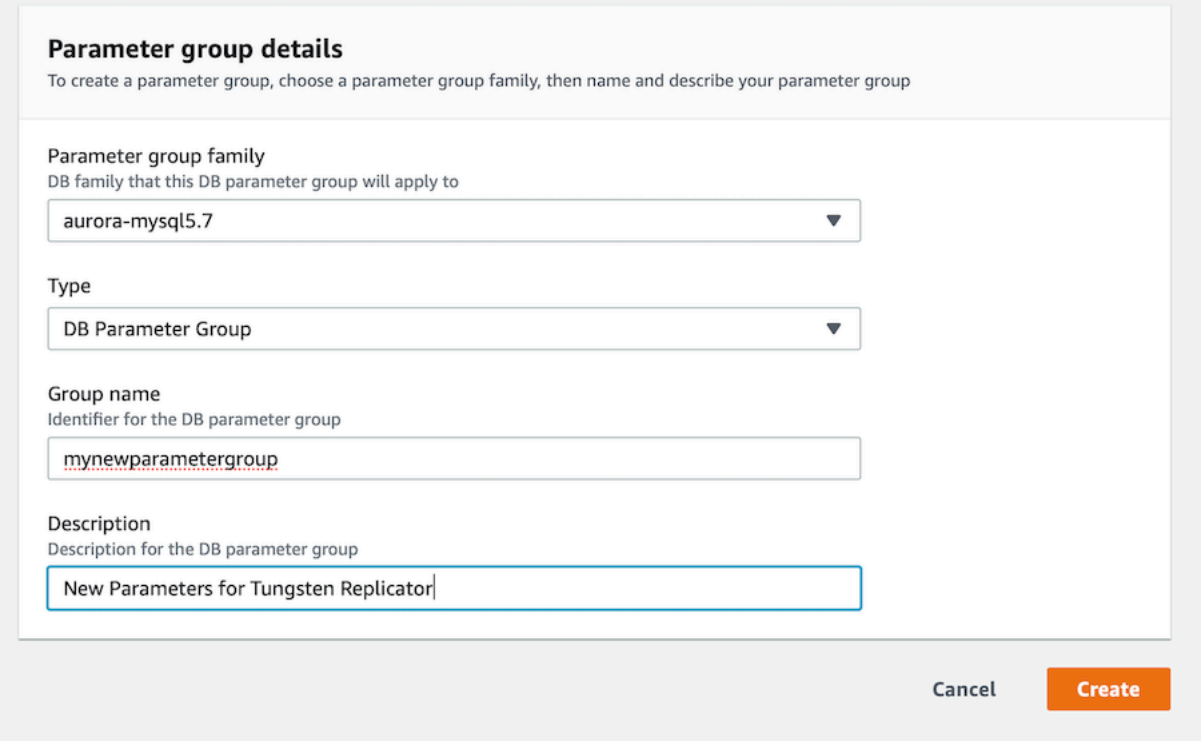
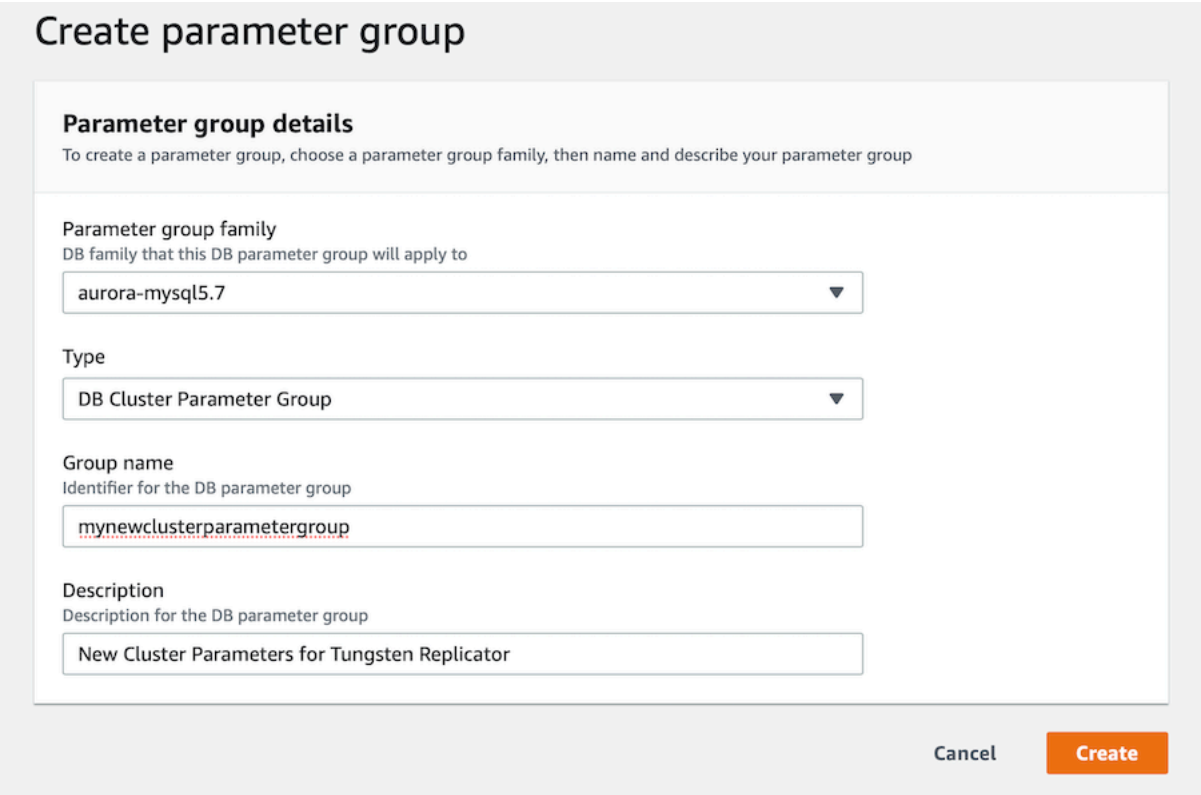



Figure 3.7. Fig 5. AWS Config



4. Now you have the two groups, you can modify the parameters accordingly, by selecting the group in the list and then selecting the "Edit" option.

Figure 3.8. Fig 6. AWS Config

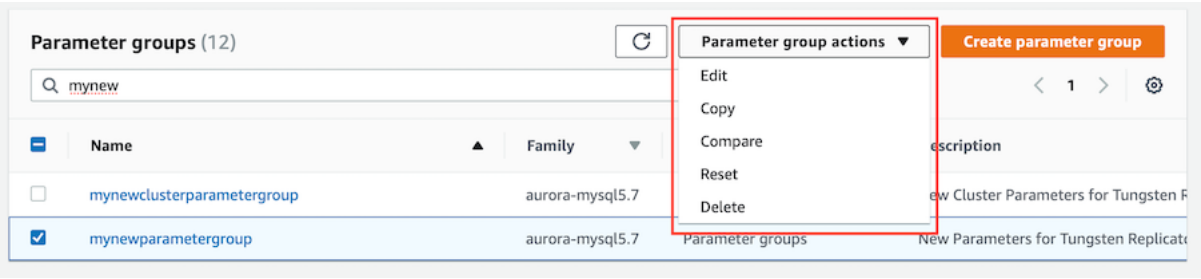
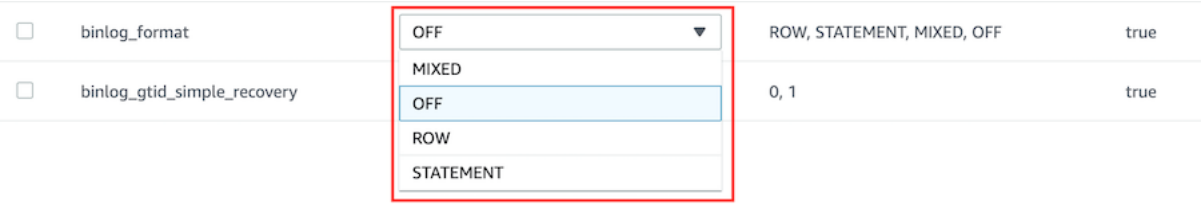



Figure 3.9. Fig 7. AWS Config



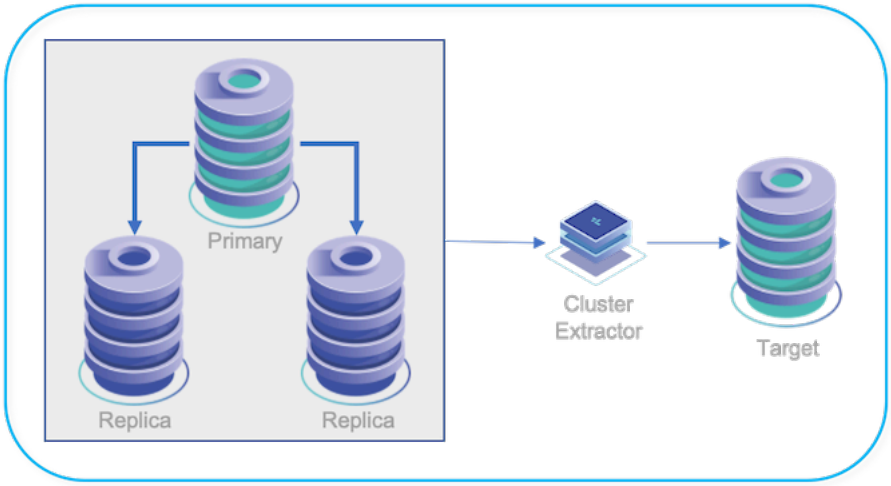

5. Now the groups are setup, you can assign these groups to existing Aurora Instances, or you can assign them during instance creation. If you are assigning to existing instances, you may need to restart the instance for certain parameters to take effect.

Some parameters can only be set via the cluster parameter group - such as enabling binary logging, others can only be change in the DB Parameter group.

## 3.4. Replicating Data Out of a Cluster

If you have an existing cluster and you want to replicate the data out to a separate standalone server using Tungsten Replicator then you can create a cluster alias, and use a Primary/Replica topology to replicate from the cluster. This allows for THL events from the cluster to be applied to a separate server for the purposes of backup or separate analysis.

Figure 3.10. Topologies: Replicating Data Out of a Cluster




During the installation process a `cluster-alias` and `cluster-slave` are declared. The `cluster-alias` describes all of the servers in the cluster and how they may be reached. The `cluster-slave` defines one or more servers that will replicate from the cluster.

The Tungsten Replicator will be installed on the Cluster-Extractor server. That server will download THL data and apply them to the local server. If the Cluster-Extractor has more than one server; one of them will be declared the relay (or Primary). The other members of the Cluster-Extractor may also download THL data from that server.

If the relay for the Cluster-Extractor fails; the other nodes will automatically start downloading THL data from a server in the cluster. If a non-relay server fails; it will not have any impact on the other members.

## 3.4.1. Prepare: Replicating Data Out of a Cluster

1. Identify the cluster to replicate from. You will need the Primary, Replicas and THL port (if specified). Use tpm reverse from a cluster member to find the correct values.
2. If you are replicating to a non-MySQL server. Update the configuration of the cluster to include the following properties prior to beginning.

```
svc-extractor-filters=colnames,pkey
property=replicator.filter.pkey.addColumnsToDeletes=true
property=replicator.filter.pkey.addPkeyToInserts=true
```

3. Identify all servers that will replicate from the cluster. If there is more than one, a relay server should be identified to replicate from the cluster and provide THL data to other servers.
4. Prepare each server according to the prerequisites for the DBMS platform it is serving. If you are working with multiple DBMS platforms; treat each platform as a different Cluster-Extractor during deployment.
5. Make sure the THL port for the cluster is open between all servers.

## 3.4.2. Deploy: Replicating Data Out of a Cluster

1. Install the Tungsten Replicator package or download the Tungsten Replicator tarball, and unpack it:

```
shell> cd /opt/continuent/software
shell> tar zxf tungsten-replicator-7.1.2-81.tar.gz
```

2. Change to the unpackaged directory:

```
shell> cd tungsten-replicator-7.1.2-81
```

3. Configure the replicator

   Click the link below to switch examples between Staging and INI methods

   Show Staging

   Show INI

```
shell> ./tools/tpm configure defaults \
    --install-directory=/opt/continuent \
    --profile-script=~/.bash_profile \
    --replication-password=secret \
    --replication-port=13306 \
    --replication-user=tungsten \
    --user=tungsten \
    --rest-api-admin-user=apiuser \
    --rest-api-admin-pass=secret

shell> ./tools/tpm configure alpha \
    --master=host1 \
    --slaves=host2,host3 \
    --thl-port=2112 \
    --topology=cluster-alias

shell> ./tools/tpm configure beta \
    --relay=host6 \
    --relay-source=alpha \
    --topology=cluster-slave
```

```
shell> vi /etc/tungsten/tungsten.ini
```

```
[defaults]
install-directory=/opt/continuent
profile-script=~/.bash_profile
replication-password=secret
replication-port=13306
replication-user=tungsten
user=tungsten
rest-api-admin-user=apiuser
rest-api-admin-pass=secret

[alpha]
master=host1
slaves=host2,host3
```

```
thl-port=2112
topology=cluster-alias

[beta]
relay=host6
relay-source=alpha
topology=cluster-slave
```

Configuration group **defaults**

The description of each of the options is shown below; click the icon to hide this detail:

Click the icon to show a detailed description of each argument.

- `--install-directory=/opt/continuent` [409]

  `install-directory=/opt/continuent` [409]

  Path to the directory where the active deployment will be installed. The configured directory will contain the software, THL and relay log information unless configured otherwise.

- `--profile-script=~/.bash_profile` [417]

  `profile-script=~/.bash_profile` [417]

  Append commands to include env.sh in this profile script

- `--replication-password=secret` [420]

  `replication-password=secret` [420]

  The password to be used when connecting to the database using the corresponding `--replication-user` [421].

- `--replication-port=13306` [421]

  `replication-port=13306` [421]

  The network port used to connect to the database server. The default port used depends on the database being configured.

- `--replication-user=tungsten` [421]

  `replication-user=tungsten` [421]

  For databases that required authentication, the username to use when connecting to the database using the corresponding connection method (native, JDBC, etc.).

- `--user=tungsten` [429]

  `user=tungsten` [429]

  System User

- `--rest-api-admin-user=apiuser` [422]

  `rest-api-admin-user=apiuser` [422]
  Optional: Must be specified along with rest-api-admin-pass if you wish to access the full API features and use the Dashboard GUI for cluster installations.

- `--rest-api-admin-pass=secret` [422]

  `rest-api-admin-pass=secret` [422]
  Optional: Must be specified along with rest-api-admin-user if you wish to access the full API features.

Configuration group **alpha**

The description of each of the options is shown below; click the icon to hide this detail:

Click the icon to show a detailed description of each argument.

- `--master=host1` [412]

  `master=host1` [412]

The hostname of the primary (extractor) within the current service.

- `--slaves=host2,host3` [423]

  `slaves=host2,host3` [423]

  What are the Replicas for this dataservice?

- `--thl-port=2112` [429]

  `thl-port=2112` [429]

  Port to use for THL Operations

- `--topology=cluster-alias` [429]

  `topology=cluster-alias` [429]

  Replication topology for the dataservice.

Configuration group **beta**

The description of each of the options is shown below; click the icon to hide this detail:

Click the icon to show a detailed description of each argument.

- `--relay=host6` [412]

  `relay=host6` [412]

  The hostname of the primary (extractor) within the current service.

- `--relay-source=alpha` [419]

  `relay-source=alpha` [419]

  Dataservice name to use as a relay source

- `--topology=cluster-slave` [429]

  `topology=cluster-slave` [429]

  Replication topology for the dataservice.

### Important

If you are replicating to a non-MySQL server. Include the following steps in your configuration.

```
shell> mkdir -p /opt/continuent/share/
shell> cp tungsten-replicator/support/filters-config/convertstringfrommysql.json »
   /opt/continuent/share/
```

Then, include the following parameters in the configuration

```
property=replicator.stage.remote-to-thl.filters=convertstringfrommysql
property=replicator.filter.convertstringfrommysql.definitionsFile= »
   /opt/continuent/share/convertstringfrommysql.json
```

### Important

This dataservice `cluster-alias` name MUST be the same as the cluster dataservice name that you are replicating from.

### Note

Do not include `start-and-report=true` [424] if you are taking over for MySQL native replication. See Section 7.10.1, “Migrating from MySQL Native Replication 'In-Place'” for next steps after completing installation.

4. Once the configuration has been completed, you can perform the installation to set up the services using this configuration:

```
shell> ./tools/tpm install
```

During the installation and startup, tpm will notify you of any problems that need to be fixed before the service can be correctly installed and started. If the service starts correctly, you should see the configuration and current status of the service.

If the installation process fails, check the output of the `/tmp/tungsten-configure.log` file for more information about the root cause.

The cluster should be installed and ready to use.

# Chapter 4. Deploying Appliers

The following sections outline the steps to configure the replicator for applying into your target of choice. Each section covers the basic configuration to deploy an applier in each of the deployment models (Onboard or Offboard).

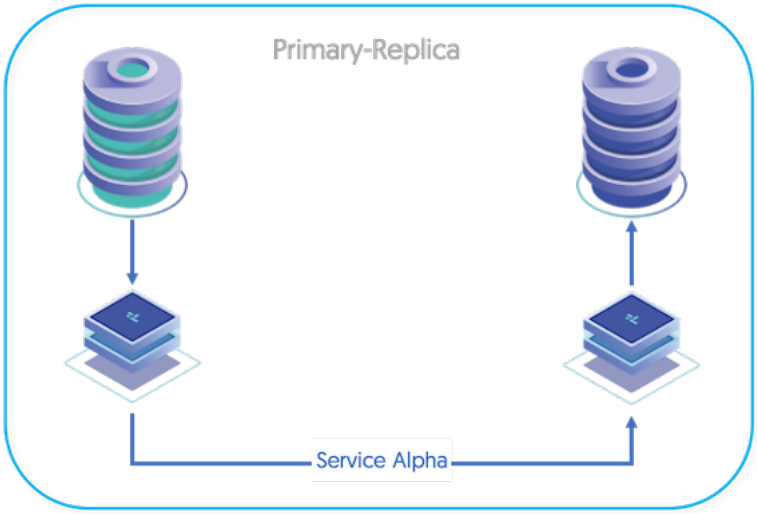
Before preparing the applier configuration, follow the steps outlined in Chapter 3, *Deploying MySQL Extractors* to configure the extractor.

## 4.1. Deploying the MySQL Applier

Deploying the MySQL applier is the most straight forward of deployments. This section covers configuration of the applier into all releases of MySQL, including Amazon RDS, Amazon Aurora, Google Cloud SQL and Microsoft Azure.

- Service Alpha on host1 extracts the information from the MySQL binary log into THL.
- Service Alpha reads the information from the remote replicator as THL, and applies that to the target MySQL instance via a JDBC Connector.

Figure 4.1. Topologies: Replicating to MySQL




The Applier replicator can be installed on:

- A host with write access to the target database host
- An EC2 Host with write access to the target Instance
- The same host as the target database
- The same host as the extractor (See Section 5.3, "Deploying Multiple Replicators on a Single Host")

### 4.1.1. Preparing for MySQL Replication

Configure the source and target hosts following the prerequisites outlined in Appendix B, *Prerequisites* then follow the appropriate steps for the required extractor topology outlined in Chapter 3, *Deploying MySQL Extractors*.

- MySQL Target

  Applies to:

  - Standalone hosted instances
  - EC2 hosted instances
  - Google Cloud hosted instances
  - Microsoft Azure hosted instances

  To prepare the target MySQL Database, ensure the user accounts are created as per the steps outlined in Section B.4.5, "MySQL User Configuration"

- Amazon RDS/Amazon Aurora Target

  For Amazon based targets, as we do not have access to the host, nor can we configure accounts with elevated privileges, follow the steps in Section B.4.6, "MySQL Unprivileged Users" to prepare the target for replication

The data replicated from MySQL can be any data, although there are some known limitations and assumptions made on the way the information is transferred.

- Table format should be updated to UTF8 by updating the MySQL configuration (`my.cnf`):

```
character-set-server=utf8
collation-server=utf8_general_ci
```

- To prevent timezone configuration storing zone adjusted values and exporting this information to the binary log and AmazonRDS, fix the timezone configuration to use UTC within the configuration file (`my.cnf`):

```
default-time-zone='+00:00'
```

If your target is an Amazon RDS or Aurora Instance, that has not yet been created, follow the steps in Section 4.1.2, "Prepare Amazon RDS/Amazon Aurora"

If your target is a hosted MySQL environment, proceed to Section 4.1.3, "Install MySQL Applier"

## 4.1.2. Prepare Amazon RDS/Amazon Aurora

- Create the Amazon Instance

  If the instance does not already exist, create the Amazon RDS or Amazon Aurora instance and take a note of the endpoint URL reported. This information will be required when configuring the replicator service.

  Also take a note of the user and password used for connecting to the instance.

- Check your security group configuration.

  The host used as the Target for applying changes to the Amazon instance must have been added to the security groups. Within Amazon RDS and Aurora, security groups configure the hosts that are allowed to connect to the Amazon instance, and hence update information within the database. The configuration must include the IP address of the Applier replicator, whether that host is within Amazon EC2 or external.

- Change RDS/Aurora instance properties

  Depending on the configuration and data to be replicated, the parameter of the running instance may need to be modified. For example, the *`max_allowed_packet`* parameter may need to be increased.

  For more information on changing parameters, see Section 3.3.1, "Changing Amazon RDS/Aurora Instance Configurations".

## 4.1.3. Install MySQL Applier

The applier will read information from the Extractor and write database changes into the target instance.

To configure the Applier replicator for either local or remote MySQL or for Amazon RDS/Aurora, the process is the same, but with a slightly different configuration, this is outlined below:

- Unpack the Tungsten Replicator distribution in staging directory:

```
shell> tar zxf tungsten-replicator-7.1.2-81.tar.gz
```

- Change into the staging directory:

```
shell> cd tungsten-replicator-7.1.2-81
```

- Use the appropriate template config for your target
  - Section 4.1.3.1, "Local and Remote MySQL Targets"
  - Section 4.1.3.2, "Amazon RDS and Amazon Aurora Targets"
- **Note**

  > If you plan to make full use of the REST API (which is enabled by default) you will need to also configure a username and password for API access. This must be done by specifying the following options in your configuration:

```
rest-api-admin-user=tungsten
rest-api-admin-pass=secret
```

- Once the prerequisites and configuring of the installation has been completed, the software can be installed:

```
shell> ./tools/tpm install
```

If the installation process fails, check the output of the `/tmp/tungsten-configure.log` file for more information about the root cause.

The replicators can now be started using the replicator command.

The status of the replicator can be checked and monitored by using the trepctl command.

### 4.1.3.1. Local and Remote MySQL Targets

- Configure the installation using tpm:

  Show Staging

  Show INI

```
shell> ./tools/tpm configure defaults \
    --reset \
    --install-directory=/opt/continuent \
    --user=tungsten \
    --mysql-allow-intensive-checks=true \
    --profile-script=~/.bash_profile \
    --rest-api-admin-user=apiuser \
    --rest-api-admin-pass=secret

shell> ./tools/tpm configure alpha \
    --master=sourcehost \
    --members=localhost,sourcehost \
    --datasource-type=mysql \
    --replication-user=tungsten \
    --replication-password=secret \
    --replication-host=remotedbhost
```

```
shell> vi /etc/tungsten/tungsten.ini
```

```
[defaults]
install-directory=/opt/continuent
user=tungsten
mysql-allow-intensive-checks=true
profile-script=~/.bash_profile
rest-api-admin-user=apiuser
rest-api-admin-pass=secret

[alpha]
master=sourcehost
members=localhost,sourcehost
datasource-type=mysql
replication-user=tungsten
replication-password=secret
replication-host=remotedbhost
```

  Configuration group **`defaults`**

  The description of each of the options is shown below; click the icon to hide this detail:

  Click the icon to show a detailed description of each argument.

  - `--reset` [422]

    `reset` [422]

    For staging configurations, deletes all pre-existing configuration information between updating with the new configuration values.

  - `--install-directory=/opt/continuent` [409]

    `install-directory=/opt/continuent` [409]

    Path to the directory where the active deployment will be installed. The configured directory will contain the software, THL and relay log information unless configured otherwise.

  - `--user=tungsten` [429]

`user=tungsten` [429]

System User

- `--mysql-allow-intensive-checks=true` [413]

  `mysql-allow-intensive-checks=true` [413]

  For MySQL installation, enables detailed checks on the supported data types within the MySQL database to confirm compatibility. This includes checking each table definition individually for any unsupported data types.

- `--profile-script=~/.bash_profile` [417]

  `profile-script=~/.bash_profile` [417]

  Append commands to include env.sh in this profile script

- `--rest-api-admin-user=apiuser` [422]

  `rest-api-admin-user=apiuser` [422]
  Optional: Must be specified along with rest-api-admin-pass if you wish to access the full API features and use the Dashboard GUI for cluster installations.

- `--rest-api-admin-pass=secret` [422]

  `rest-api-admin-pass=secret` [422]
  Optional: Must be specified along with rest-api-admin-user if you wish to access the full API features.

Configuration group **`alpha`**

The description of each of the options is shown below; click the icon to hide this detail:

Click the icon to show a detailed description of each argument.

- `--master=sourcehost` [412]

  `master=sourcehost` [412]

  The hostname of the primary (extractor) within the current service.

- `--members=localhost,sourcehost` [413]

  `members=localhost,sourcehost` [413]

  Hostnames for the dataservice members

- `--datasource-type=mysql` [402]

  `datasource-type=mysql` [402]

  Database type

- `--replication-user=tungsten` [421]

  `replication-user=tungsten` [421]

  For databases that required authentication, the username to use when connecting to the database using the corresponding connection method (native, JDBC, etc.).

- `--replication-password=secret` [420]

  `replication-password=secret` [420]

  The password to be used when connecting to the database using the corresponding `--replication-user` [421].

- `--replication-host=remotedbhost` [420]

  `replication-host=remotedbhost` [420]

  Hostname of the datasource where the database is located. If the specified hostname matches the current host or member name, the database is assumed to be local. If the hostnames do not match, extraction is assumed to be via remote access. For MySQL hosts, this configures a remote replication Replica (relay) connection.

`replication-host` [420] should only be added to the above configuration if the target MySQL Database is on a different host to the applier installation

### 4.1.3.2. Amazon RDS and Amazon Aurora Targets

- Amazon RDS and Amazon Aurora Targets

  Configure the installation using tpm:

  Show Staging

  Show INI

```
shell> ./tools/tpm configure defaults \
    --reset \
    --install-directory=/opt/continuent \
    --user=tungsten \
    --mysql-allow-intensive-checks=true \
    --profile-script=~/.bash_profile \
    --skip-validation-check=InstallerMasterSlaveCheck \
    --skip-validation-check=MySQLPermissionsCheck \
    --skip-validation-check=MySQLBinaryLogsEnabledCheck \
    --skip-validation-check=MySQLMyISAMCheck \
    --skip-validation-check=RowBasedBinaryLoggingCheck \
    --rest-api-admin-user=apiuser \
    --rest-api-admin-pass=secret

shell> ./tools/tpm configure alpha \
    --master=sourcehost \
    --members=localhost,sourcehost \
    --datasource-type=mysql \
    --datasource-mysql-conf=/dev/null \
    --replication-user=rdsuser \
    --replication-password=secret \
    --privileged-slave=false \
    --replication-host=rds-endpoint-url \
    --service-type=remote
```

```
shell> vi /etc/tungsten/tungsten.ini
```

```
[defaults]
install-directory=/opt/continuent
user=tungsten
mysql-allow-intensive-checks=true
profile-script=~/.bash_profile
skip-validation-check=InstallerMasterSlaveCheck
skip-validation-check=MySQLPermissionsCheck
skip-validation-check=MySQLBinaryLogsEnabledCheck
skip-validation-check=MySQLMyISAMCheck
skip-validation-check=RowBasedBinaryLoggingCheck
rest-api-admin-user=apiuser
rest-api-admin-pass=secret

[alpha]
master=sourcehost
members=localhost,sourcehost
datasource-type=mysql
datasource-mysql-conf=/dev/null
replication-user=rdsuser
replication-password=secret
privileged-slave=false
replication-host=rds-endpoint-url
service-type=remote
```

  Configuration group **`defaults`**

  The description of each of the options is shown below; click the icon to hide this detail:

  Click the icon to show a detailed description of each argument.

- `--reset` [422]

  `reset` [422]

  For staging configurations, deletes all pre-existing configuration information between updating with the new configuration values.

- `--install-directory=/opt/continuent` [409]

`install-directory=/opt/continuent` [409]

Path to the directory where the active deployment will be installed. The configured directory will contain the software, THL and relay log information unless configured otherwise.

- `--user=tungsten` [429]

  `user=tungsten` [429]

  System User

- `--mysql-allow-intensive-checks=true` [413]

  `mysql-allow-intensive-checks=true` [413]

  For MySQL installation, enables detailed checks on the supported data types within the MySQL database to confirm compatibility. This includes checking each table definition individually for any unsupported data types.

- `--profile-script=~/.bash_profile` [417]

  `profile-script=~/.bash_profile` [417]

  Append commands to include env.sh in this profile script

- `--skip-validation-check=InstallerMasterSlaveCheck` [369]

  `skip-validation-check=InstallerMasterSlaveCheck` [369]

  The `--skip-validation-check` [369] disables a given validation check. If any validation check fails, the installation, validation or configuration will automatically stop.

  > **Warning**
  >
  > Using this option enables you to bypass the specified check, although skipping a check may lead to an invalid or non-working configuration.

  You can identify a given check if an error or warning has been raised during configuration. For example, the default table type check:

  ```
  ...
  ERROR >> centos >> The datasource root@centos:3306 (WITH PASSWORD) »
   uses MyISAM as the default storage engine (MySQLDefaultTableTypeCheck)
  ...
  ```

  The check in this case is `MySQLDefaultTableTypeCheck` [380], and could be ignored using `--skip-validation-check=MySQLDefaultTableTypeCheck` [369].

  Setting both `--skip-validation-check` [369] and `--enable-validation-check` [367] is equivalent to explicitly disabling the specified check.

- `--skip-validation-check=MySQLPermissionsCheck` [369]

  `skip-validation-check=MySQLPermissionsCheck` [369]

  The `--skip-validation-check` [369] disables a given validation check. If any validation check fails, the installation, validation or configuration will automatically stop.

  > **Warning**
  >
  > Using this option enables you to bypass the specified check, although skipping a check may lead to an invalid or non-working configuration.

  You can identify a given check if an error or warning has been raised during configuration. For example, the default table type check:

  ```
  ...
  ERROR >> centos >> The datasource root@centos:3306 (WITH PASSWORD) »
   uses MyISAM as the default storage engine (MySQLDefaultTableTypeCheck)
  ...
  ```

  The check in this case is `MySQLDefaultTableTypeCheck` [380], and could be ignored using `--skip-validation-check=MySQLDefaultTableTypeCheck` [369].

  Setting both `--skip-validation-check` [369] and `--enable-validation-check` [367] is equivalent to explicitly disabling the specified check.

- `--skip-validation-check=MySQLBinaryLogsEnabledCheck` [369]

  `skip-validation-check=MySQLBinaryLogsEnabledCheck` [369]

  The `--skip-validation-check` [369] disables a given validation check. If any validation check fails, the installation, validation or configuration will automatically stop.

  > **Warning**
  >
  > Using this option enables you to bypass the specified check, although skipping a check may lead to an invalid or non-working configuration.

  You can identify a given check if an error or warning has been raised during configuration. For example, the default table type check:

  ```
  ...
  ERROR >> centos >> The datasource root@centos:3306 (WITH PASSWORD) »
   uses MyISAM as the default storage engine (MySQLDefaultTableTypeCheck)
  ...
  ```

  The check in this case is `MySQLDefaultTableTypeCheck` [380], and could be ignored using `--skip-validation-check=MySQLDefaultTableTypeCheck` [369].

  Setting both `--skip-validation-check` [369] and `--enable-validation-check` [367] is equivalent to explicitly disabling the specified check.

- `--skip-validation-check=MySQLMyISAMCheck` [369]

  `skip-validation-check=MySQLMyISAMCheck` [369]

  The `--skip-validation-check` [369] disables a given validation check. If any validation check fails, the installation, validation or configuration will automatically stop.

  > **Warning**
  >
  > Using this option enables you to bypass the specified check, although skipping a check may lead to an invalid or non-working configuration.

  You can identify a given check if an error or warning has been raised during configuration. For example, the default table type check:

  ```
  ...
  ERROR >> centos >> The datasource root@centos:3306 (WITH PASSWORD) »
   uses MyISAM as the default storage engine (MySQLDefaultTableTypeCheck)
  ...
  ```

  The check in this case is `MySQLDefaultTableTypeCheck` [380], and could be ignored using `--skip-validation-check=MySQLDefaultTableTypeCheck` [369].

  Setting both `--skip-validation-check` [369] and `--enable-validation-check` [367] is equivalent to explicitly disabling the specified check.

- `--skip-validation-check=RowBasedBinaryLoggingCheck` [369]

  `skip-validation-check=RowBasedBinaryLoggingCheck` [369]

  The `--skip-validation-check` [369] disables a given validation check. If any validation check fails, the installation, validation or configuration will automatically stop.

  > **Warning**
  >
  > Using this option enables you to bypass the specified check, although skipping a check may lead to an invalid or non-working configuration.

  You can identify a given check if an error or warning has been raised during configuration. For example, the default table type check:

  ```
  ...
  ERROR >> centos >> The datasource root@centos:3306 (WITH PASSWORD) »
   uses MyISAM as the default storage engine (MySQLDefaultTableTypeCheck)
  ...
  ```

  The check in this case is `MySQLDefaultTableTypeCheck` [380], and could be ignored using `--skip-validation-check=MySQLDefaultTableTypeCheck` [369].

  Setting both `--skip-validation-check` [369] and `--enable-validation-check` [367] is equivalent to explicitly disabling the specified check.

- `--rest-api-admin-user=apiuser` [422]

  `rest-api-admin-user=apiuser` [422]
  Optional: Must be specified along with rest-api-admin-pass if you wish to access the full API features and use the Dashboard GUI for cluster installations.

- `--rest-api-admin-pass=secret` [422]

  `rest-api-admin-pass=secret` [422]
  Optional: Must be specified along with rest-api-admin-user if you wish to access the full API features.

Configuration group **`alpha`**

The description of each of the options is shown below; click the icon to hide this detail:

Click the icon to show a detailed description of each argument.

- `--master=sourcehost` [412]

  `master=sourcehost` [412]

  The hostname of the primary (extractor) within the current service.

- `--members=localhost,sourcehost` [413]

  `members=localhost,sourcehost` [413]

  Hostnames for the dataservice members

- `--datasource-type=mysql` [402]

  `datasource-type=mysql` [402]

  Database type

- `--datasource-mysql-conf=/dev/null` [401]

  `datasource-mysql-conf=/dev/null` [401]

  MySQL config file

- `--replication-user=rdsuser` [421]

  `replication-user=rdsuser` [421]

  For databases that required authentication, the username to use when connecting to the database using the corresponding connection method (native, JDBC, etc.).

- `--replication-password=secret` [420]

  `replication-password=secret` [420]

  The password to be used when connecting to the database using the corresponding `--replication-user` [421].

- `--privileged-slave=false` [417]

  `privileged-slave=false` [417]

  Does the login for the Replica database service have superuser privileges

- `--replication-host=rds-endpoint-url` [420]

  `replication-host=rds-endpoint-url` [420]

  Hostname of the datasource where the database is located. If the specified hostname matches the current host or member name, the database is assumed to be local. If the hostnames do not match, extraction is assumed to be via remote access. For MySQL hosts, this configures a remote replication Replica (relay) connection.

- `--service-type=remote` [423]

  `service-type=remote` [423]

What is the replication service type?

## 4.1.4. Management and Monitoring of MySQL Deployments

Replication to MySQL and Amazon based instances operates in the same manner as all other replication environments. The current status can be monitored using trepctl. On the Extractor:

```
shell> trepctl status
Processing status command...
NAME                     VALUE
----                     -----
appliedLastEventId     : mysql-bin.000043:0000000000000291;84
appliedLastSeqno       : 2320
appliedLatency         : 0.733
channels               : 1
clusterName            : alpha
currentEventId         : mysql-bin.000043:0000000000000291
currentTimeMillis      : 1387544952494
dataServerHost         : host1
extensions             :
host                   : host1
latestEpochNumber      : 60
masterConnectUri       : thl://localhost:/
masterListenUri        : thl://host1:2112/
maximumStoredSeqNo     : 2320
minimumStoredSeqNo     : 0
offlineRequests        : NONE
pendingError           : NONE
pendingErrorCode       : NONE
pendingErrorEventId    : NONE
pendingErrorSeqno      : -1
pendingExceptionMessage: NONE
pipelineSource         : jdbc:mysql:thin://host1:13306/
relativeLatency        : 23.494
resourcePrecedence     : 99
rmiPort                : 10000
role                   : master
seqnoType              : java.lang.Long
serviceName            : alpha
serviceType            : local
simpleServiceName      : alpha
siteName               : default
sourceId               : host1
state                  : ONLINE
timeInStateSeconds     : 99525.477
transitioningTo        :
uptimeSeconds          : 99527.364
useSSLConnection       : false
version                : Tungsten Replicator 7.1.2 build 81
Finished status command...
```

On the Applier, use trepctl and monitor the *appliedLatency* and *appliedLastSeqno*. The output will include the hostname of the Amazon RDS instance:

```
shell> trepctl status
Processing status command...
NAME                     VALUE
----                     -----
appliedLastEventId     : mysql-bin.000043:0000000000000291;84
appliedLastSeqno       : 2320
appliedLatency         : 797.615
channels               : 1
clusterName            : default
currentEventId         : NONE
currentTimeMillis      : 1387545785268
dataServerHost         : documentationtest.cnlhon44f2wq.eu-west-1.rds.amazonaws.com
extensions             :
host                   : documentationtest.cnlhon44f2wq.eu-west-1.rds.amazonaws.com
latestEpochNumber      : 60
masterConnectUri       : thl://host1:2112/
masterListenUri        : thl://host2:2112/
maximumStoredSeqNo     : 2320
minimumStoredSeqNo     : 0
offlineRequests        : NONE
pendingError           : NONE
pendingErrorCode       : NONE
pendingErrorEventId    : NONE
pendingErrorSeqno      : -1
pendingExceptionMessage: NONE
pipelineSource         : thl://host1:2112/
relativeLatency        : 856.268
```

```
resourcePrecedence     : 99
rmiPort                : 10000
role                   : slave
seqnoType              : java.lang.Long
serviceName            : alpha
serviceType            : local
simpleServiceName      : alpha
siteName               : default
sourceId               : documentationtest.cnlhon44f2wq.eu-west-1.rds.amazonaws.com
state                  : ONLINE
timeInStateSeconds     : 461.885
transitioningTo        :
uptimeSeconds          : 668.606
useSSLConnection       : false
version                : Tungsten Replicator 7.1.2 build 81
Finished status command...
```

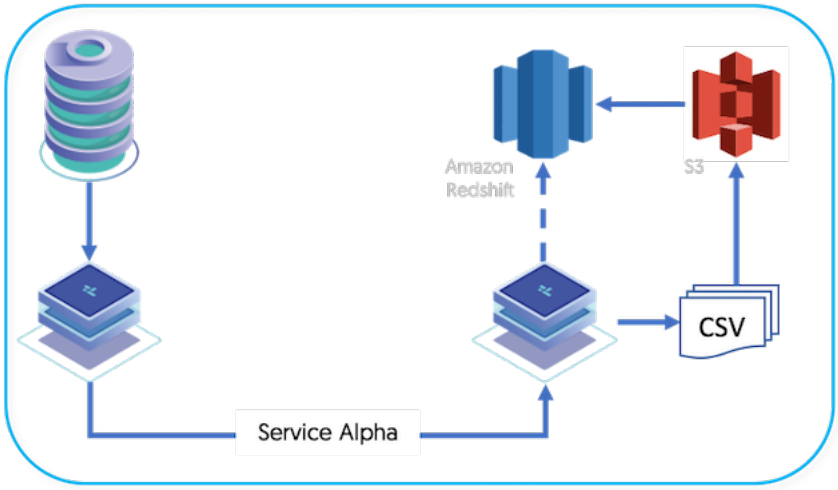
# 4.2. Deploying the Amazon Redshift Applier

Amazon Redshift is a cloud-based data warehouse service that integrates with other Amazon services, such as S3, to provide an SQL-like interface to the loaded data. Replication for Amazon Redshift moves data from MySQL datastores, through S3, and into the Redshift environment in real-time, avoiding the need to manually export and import the data.

Replication to Amazon Redshift operates as follows:

- Data is extracted from the source database into THL.
- When extracting the data from the THL, the Amazon Redshift replicator writes the data into CSV files according to the name of the source tables. The files contain all of the row-based data, including the global transaction ID generated by the extractor during replication, and the operation type (insert, delete, etc) as part of the CSV data.
- The generated CSV files are loaded into Amazon S3 using either the s3cmd command or the aws s3 cli tools. This enables easy access to your Amazon S3 installation and simplifies the loading.
- The CSV data is loaded from S3 into Redshift staging tables using the Redshift `COPY` command, which imports raw CSV into Redshift tables.
- SQL statements are then executed within Redshift to perform updates on the live version of the tables, using the CSV, batch loaded, information, deleting old rows, and inserting the new data when performing updates to work effectively within the confines of Amazon Redshift operation.

Figure 4.2. Topologies: Replicating to Amazon Redshift



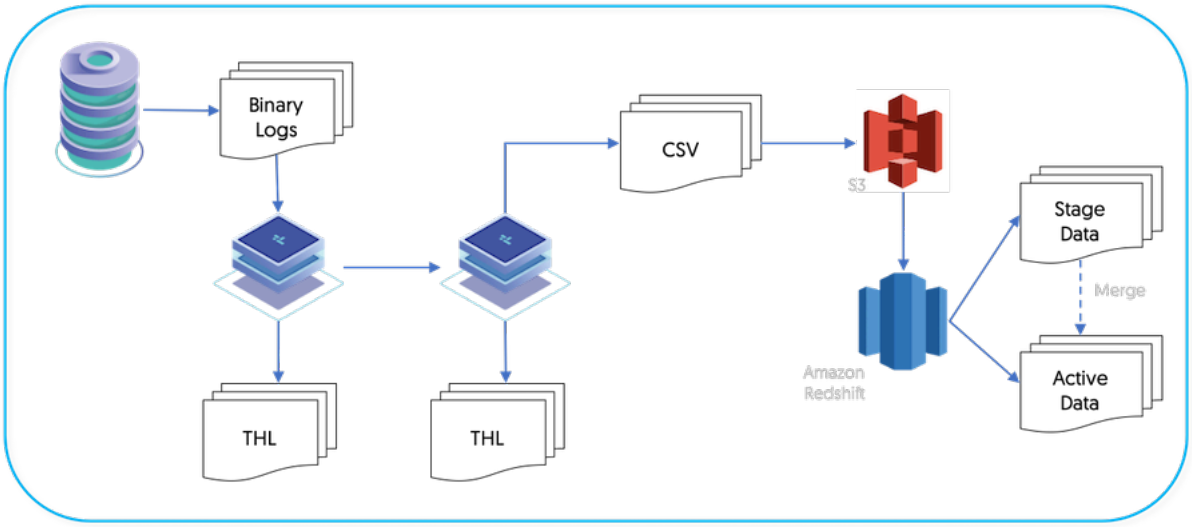

Setting up replication requires setting up both the Extractor and Applier components as two different configurations, one for MySQL and the other for Amazon Redshift. Replication also requires some additional steps to ensure that the Amazon Redshift host is ready to accept the replicated data that has been extracted. Tungsten Replicator provides all the tools required to perform these operations during the installation and setup.

## 4.2.1. Redshift Replication Operation

The Redshift applier makes use of the JavaScript based batch loading system (see Section 5.6.4, "JavaScript Batchloader Scripts"). This constructs change data from the source-database. The change data is then loaded into staging tables, at which point a process will then merge

the change data up into the base tables A summary of this basic structure can be seen in Figure 4.3, "Topologies: Redshift Replication Operation".

Figure 4.3. Topologies: Redshift Replication Operation




Different object types within the two systems are mapped as follows:

| MySQL | Redshift |
|---|---|
| Instance | Database |
| Database | Schema |
| Table | Table |

The full replication of information operates as follows:

1. Data is extracted from the source database using the standard extractor, for example by reading the row change data from the binlog in MySQL.

2. The Section 11.4.5, "ColumnName Filter" filter is used to extract column name information from the database. This enables the row-change information to be tagged with the corresponding column information. The data changes, and corresponding row names, are stored in the THL.

   The Section 11.4.31, "PrimaryKey Filter" filter is used to extract primary key data from the source tables.

3. On the Applier replicator, the THL data is read and written into batch-files in the character-separated value format.

   The information in these files is change data, and contains not only the original row values from the source tables, but also metadata about the operation performed (i.e. `INSERT`, `DELETE` or `UPDATE`, and the primary key of for each table. All `UPDATE` statements are recorded as a `DELETE` of the existing data, and an `INSERT` of the new data.

   In addition to these core operation types, the batch applier can also be configured to record `UPDATE` operations that result in `INSERT` or `DELETE` rows. This enables Redshift to process the update information more simply than performing the individual `DELETE` and `INSERT` operations.

4. A second process uses the CSV stage data and any existing data, to build a materialized view that mirrors the source table data structure.

The staging files created by the replicator are in a specific format that incorporates change and operation information in addition to the original row data.

- The format of the files is a character separated values file, with each row separated by a newline, and individual fields separated by the character `0x01`. This is supported by Hive as a native value separator.

- The content of the file consists of the full row data extracted from the Source, plus metadata describing the operation for each row, the sequence number, and then the full row information.

| Operation | Sequence No | Table-specific primary key | DateTime | Table-columns... |
|---|---|---|---|---|
| OPTYPE | *SEQNO* [549] that generated this row | PRIMARYKEY | DATATIME of source table commit | |

The operation field will match one of the following values

| Operation | Description | Notes |
|---|---|---|
| I | Row is an `INSERT` of new data | |
| D | Row is `DELETE` of existing data | |
| UI | Row is an `UPDATE` which caused `INSERT` of data | |
| UD | Row is an `UPDATE` which caused `DELETE` of data | |

For example, the MySQL row from an `INSERT` of:

```
|  3 | #1 Single | 2006 | Cats and Dogs (#1.4)         |
```

Is represented within the CSV staging files generated as:

```
"I","5","3","2014-07-31 14:29:17.000","3","#1 Single","2006","Cats and Dogs (#1.4)"
```

The character separator, and whether to use quoting, are configurable within the replicator when it is deployed. For Redshift, the default behavior is to generate quoted and comma separated fields.

## 4.2.2. Preparing for Amazon Redshift Replication

Preparing the hosts for the replication process requires setting some key configuration parameters within the MySQL server to ensure that data is stored and written correctly. On the Amazon Redshift side, the database and schema must be created using the existing schema definition so that the databases and tables exist within Amazon Redshift.

Source Host

Configure the source and target hosts following the prerequisites outlined in Appendix B, *Prerequisites* then follow the appropriate steps for the required extractor topology outlined in Chapter 3, *Deploying MySQL Extractors*.

The following are *required* for replication to Amazon Redshift:

### 4.2.2.1. Redshift Preparation for Amazon Redshift Deployments

On the Amazon Redshift host, you need to perform some preparation of the destination database, first creating the database, and then creating the tables that are to be replicated. Setting up this process requires the configuration of a number of components outside of Tungsten Replicator in order to support the loading.

- An existing Amazon Web Services (AWS) account, and either the AWS Access Key and Secret Key, or configured IAM Roles, required to interact with the account through the API. For information on creating IAM Roles, see Section 4.2.2.2, "Configuring Identity Access Management within AWS"
- A configured Amazon S3 service. If the S3 service has not already been configured, visit the AWS console and sign up for the Amazon S3 service.
- The s3cmd or the aws tools installed and configured. The s3cmd can be downloaded from s3cmd on s3tools.org.

  If using the s3cmd, you should then configure the command to automatically connect to the Amazon S3 service without requiring further authentication, the `.s3cfg` in the `tungsten` users home directory should be configured as follows:

  - Using Access Keys:

    ```
    [default]
    access_key = ACCESS_KEY
    secret_key = SECRET_KEY
    ```

  - Using IAM Roles: Leave values blank - copy example as is

    ```
    [default]
    access_key =
    secret_key =
    security_token =
    ```

- Create an S3 bucket that will be used to hold the CSV files that are generated by the replicator. This can be achieved either through the web interface, or via the command-line, for example:

```
shell> s3cmd mb s3://tungsten-csv
```

- A running Redshift instance must be available, and the port and IP address of the Tungsten Cluster that will be replicating into Redshift must have been added to the Redshift instance security credentials.

  Make a note of the user and password that has been provided with access to the Redshift instance, as these will be needed when installing the applier. Also make a note of the Redshift instance address, as this will need to be provided to the applier configuration.

- Create an `s3-config-servicename.json` file based on the sample provided within `cluster-home/samples/conf/s3-config-servicename.json` within the Tungsten Replicator staging directory, or using the example below.

  Once created, the file will be copied into the `/opt/continuent/share` directory to be used by the batch applier script.

  If multiple services are being created, one file must be created for each service.

  The following example shows the use of Access and Secret Keys:

```
{
  "awsS3Path" : "s3://your-bucket-for-redshift/redshift-test",
  "awsAccessKey" : "access-key-id",
  "awsSecretKey" : "secret-access-key",
  "cleanUpS3Files" : "true"
}
```

  The following example shows the use of IAM Roles:

```
{
  "awsS3Path" : "s3://your-bucket-for-redshift/redshift-test",
  "awsIAMRole" : "arn:iam-role",
  "cleanUpS3Files" : "true"
}
```

  The allowed options for this file are as follows:

  - `awsS3Path` — the location within your S3 storage where files should be loaded.
  - `awsAccessKey` — the S3 access key to access your S3 storage. Not required if awsIAMRole is used.
  - `awsSecretKey` — the S3 secret key associated with the Access Key. Not required if awsIAMRole is used.
  - `awsIAMRole` — the IAM role configured to allow Redshift to interact with S3. Not required if awsAccessKey and awsSecretKey are in use.
  - `multiServiceTarget` (true/false) — to indicate if there are multiple appliers writing into the single Redshift Target, for example when the source is Tungsten Cluster Composite Active/Active or a Tungsten Replicator Fan-In Topology (Default: false).
  - `singleLockTable` (true/false) — to indicate the table lock behaviour when multiServiceTarget is true. Will be ignored if multiServiceTarget set to false (Default: true)
  - `lockTablePrefix` — the prefix for the lock tables when singleLockTable is false. (Default: lock_xxx_)
  - `s3Binary` — the binary to use for loading csv file up to S3. (Valid Values: s3cmd, s4cmd, aws) (Default: s3cmd)
  - `redshiftCopyOptions` — allows the passing of additional valid syntax to be added to the Redshift COPY command during csv loading from S3 into Redshift Staging Tables.

    A list of valid parameters can be found in the Redshift documentation

  - `cleanUpS3Files` — a boolean value used to identify whether the CSV files loaded into S3 should be deleted after they have been imported and merged. If set to true, the files are automatically deleted once the files have been successfully imported into the Redshift staging tables. If set to false, files are not automatically removed.
  - `gzipS3Files` — setting to true will result in the csv files being gzipped prior to loading into S3 (Default: false)
  - `storeCDCIn` — a definition table that stores the change data from the load, in addition to importing to staging and base tables. The `{schema}` and `{table}` variables will be automatically replaced with the corresponding schema and table name. For more information on keeping CDC information, see Section 4.2.5, “Keeping CDC Information”.

### 4.2.2.2. Configuring Identity Access Management within AWS

Identity Management with AWS is complex, but a useful and secure way of restriciting services interacting with each other, and for restricting user access to the AWS platform.

Tungsten Replicator for Redshift, and Tungsten Replicator for S3, requires a certain level of interaction between the replicator and S3 and between Redshift and S3.

If configuring the Tungsten Replicator for S3, you only need to follow the relevant steps to allow the replicator to access, and write to, the S3 bucket that you create.

> **Note**
>
> All versions up to and including Tungsten Replicator version 6.0 can utilise IAM Roles for uploading the csv files to S3, however for loading the data from S3 into Redshift, the only option is to use Access and Secret Keys.
>
> Tungsten Replicator version 6.1 onwards will also allow for the use of IAM Roles for loading data from S3 into Redshift.

To use IAM Roles with Tungsten Replicator you will need to create two roles, with the following recommended policies:

To allow csv files to be loaded upto S3:

- Role should be associated with the AWS Service: EC2
- AWS Defined Policy Name: AmazonS3FullAccess, or
- Define and create your own policy, with, at minimum, the ability to write to the bucket you intend to use for the Redshift Applier
- Associate this role to the EC2 instance running the Tungsten Replicator software

For use by Redshift COPY command to load csv into staging tables:

- Role should be associated with the AWS Service: Redshift
- AWS Defined Policy Name: AmazonS3FullAccess, or
- Define and create your own policy, with, at minimum, the ability to read from the bucket you intend to use for the Redshift Applier
- Associate this role to the Redshift Cluster.

> **Note**
>
> For more details and full instructions on creating and managing IAM roles, review the AWS documentation

### 4.2.2.3. Amazon Redshift DDL Generation for Amazon Redshift Deployments

In order for the data to be written into the Redshift tables, the tables must be generated. Tungsten Replicator does not replicate the DDL statements between the source and applier between heterogeneous deployments due to differences in the format of the DDL statements. The supplied ddlscan tool can translate the DDL from the source database into suitable DDL for the target database.

For each database being replicated, DDL must be generated twice, once for the staging tables where the change data is loaded, and again for the live tables. To generate the necessary DDL:

1. To generate the staging table DDL, ddlscan must be executed on the Extractor host. After the replicator has been installed, the ddlscan can automatically pick up the configuration to connect to the host, or it can be specified on the command line:

   On the source host for each database that is being replicated, run ddlscan using the `ddl-mysql-redshift-staging.vm`:

   ```
   shell> ddlscan -db test -template ddl-mysql-redshift-staging.vm
   DROP TABLE stage_xxx_test.stage_xxx_msg;
   CREATE TABLE stage_xxx_test.stage_xxx_msg
   (
     tungsten_opcode CHAR(2),
     tungsten_seqno INT,
     tungsten_row_id INT,
     tungsten_commit_timestamp TIMESTAMP,
     id INT,
     msg CHAR(80),
     PRIMARY KEY (tungsten_opcode, tungsten_seqno, tungsten_row_id)
   );
   ```

   Check the output to ensure that no errors have been generated during the process. These may indicate datatype limitations that should be identified before continuing. The generated output should be captured and then executed on the Redshift host to create the table.

2. Once the staging tables have been created, execute ddlscan again using the base table template, `ddl-mysql-redshift.vm`:

   ```
   shell> ddlscan -db test -template ddl-mysql-redshift.vm
   DROP TABLE test.msg;
   CREATE TABLE test.msg
   ```

```
(
  id INT,
  msg CHAR(80),
  PRIMARY KEY (id)
);
```

Once again, check the output for errors, then capture the output and execute the generated DDL against the Redshift instance.

The DDL templates translate datatypes as directly as possible, with the following caveats:

- The length of MySQL `VARCHAR` length is quadrupled, because MySQL counts characters, while Redshift counts bytes.
- There is no `TIME` datatype in Redshift, instead, `TIME` columns are converted to `VARCHAR(17)`.
- Primary keys from MySQL are applied into Redshift where possible.

Once the DDL has been generated within the Redshift instance, the replicator will be ready to be installed.

## 4.2.2.4. Handling Concurrent Writes from Multiple Appliers

The features outlined in this section where specifically introduced in Tungsten Replicator 6.1.4.

Redshift only supports a `SERIALIZABLE` transaction isolation level, which differs from relational databases like MySQL, which is `REPEATABLE READ` by default. Isolation Levels determine the behaviour of the database for concurrent access to the tables within transactions.

When loading data into Redshift, from multiple appliers, this isolation level can cause locking issues that would manifest as errors in the Replicator Log similiar to the following:

```
Detail: Serializable isolation violation on table - 150379, transactions forming the cycle are: 2356786, 2356787
» (pid:17914) (../../tungsten-replicator//appliers/batch/redshift.js#219)
```

In some cases, the replicator will simply retry and carry on successfully, but on very busy systems this can sometimes cause the replicator to fall back into an `OFFLINE:ERROR` state and manual intervention would be required.

To overcome this problem, the first step is to ensure that each applier has its own set of staging tables that the CSV files are loaded into. By default all staging tables will be named with the prefix `stage_xxx_`

First of all, to generate the staging tables, you would typically use ddlscan that would look something like the following:

```
shel> ddlscan -user tungsten -pass secret -url jdbc:mysql:thin://db01:3306/
  » -db hr -template ddl-mysql-redshift-staging.vm > staging.sql
```

To change the default prefix of the staging table, for example, to `stage_nyc_` you can provide the option to the ddlscan command as follows:

```
shel> ddlscan -user tungsten -pass secret -url jdbc:mysql:thin://db01:3306/
  » -db hr -template ddl-mysql-redshift-staging.vm -opt tablePrefix stage_nyc_ > staging.sql
```

You would need to execute this for each applier, changing the prefix accordingly. Once this has been executed and the tables have been built in Redshift, you will then need to add the additional property to each applier to instruct which staging tables to use. The property should be added to the `tungsten.ini` file and a tpm update issued

```
property=replicator.applier.dbms.stageTablePrefix=stage_nyc_
```

### 4.2.2.4.1. Increase load rates

The first and easiest step to try and overcome the isolation errors, would be to increase the batch commit levels and the batch commit interval. Each system works differently so there is no simple calculation to find the right level. These values should be adjusted in small increments to find the right balance for your system.

Within your configuration, adjust the following two parameters:

- `svc-block-commit-size`
- `svc-block-commit-interval`

### 4.2.2.4.2. Enable Transaction Locking

Within the redshift applier, it is possible to introduce table locking. This will enable multiple appliers to process their own THL and load the transactions without impacting, or being impacted by, other appliers.

This configuration should only be used when multiple appliers are in use, however it must also be recognised that the addition of table locking could introduce latency in applying to Redshift on extremely busy systems, it could also impact client applications from reading the tables due to Redshift's isolation level. To avoid this, table locking should also include an increase in the block commit size and block commit interval properties mentioned above.

There are two types of table locking approaches, depending upon your environment will determine which approach is better for you.

- Single Lock Table: This approach should be used for appliers in extremely busy systems where a block-commit-size of 500000 or greater does not eliminate isolation errors and where mutliple tables are updated within each transaction.
- One Lock Table per Base Table: This approach should be used for appliers in less busy systems, or where parallel apply has been enabled within the applier, regardless of system activity levels.

To enable the single lock table approach:

- The following option should be added to the `s3-config-servicename.json` file:

```
"multiServiceTarget": "true"
```

- Connect to Redshift with the same account used by the applier, and using the DDL below, create the lock table:

```
CREATE TABLE public.tungsten_lock_table
(
  ID  INT
);
```

To enable the lock table per base table approach:

- The following option should be added to the `s3-config-servicename.json` file:

```
"multiServiceTarget": "true",
  "singleLockTable": "false"
```

- Create a lock table for each of the base tables within Redshift. A ddlscan template can be used to generate the ddl. In the following example the ddlscan command is generating lock table ddl for all tables within the `hr` schema:

```
shel> ddlscan -user tungsten -pass secret -url jdbc:mysql:thin://db01:3306/
  » -db hr -template ddl-mysql-redshift-lock.vm > outfile.sql
```

  Execute the output from ddlscan into redshift

After enabling either of the above methods, if replication has already been installed you will need to simply restart the replicator by issuing the following:

```
shel> replicator restart
```

## 4.2.3. Install Amazon Redshift Applier

Replication into Redshift requires two separate replicator installations, one that extracts information from the source database, and a second that generates the CSV files, loads those files into S3 and then executes the statements on the Redshift database to import the CSV data and apply the transformations to build the final tables.

The two replication services can operate on the same machine, (See Section 5.3, “Deploying Multiple Replicators on a Single Host”) or they can be installed on two different machines.

Once you have completed the configuration of the Amazon Redshift database, you can configure and install the applier as described using the steps below.

1. Before installing the applier, the following additions need adding to the extractor configuration. Apply the following parameter to the extractor configuration before installing the applier

   Add the following the `/etc/tungsten/tungsten.ini`

```
[alpha]
...Existing Replicator Config...
enable-heterogeneous-service=true

shell> tpm update
```

   **Note**

   The above step is only applicable for standalone extractors. If you are configuring replications from an existing Tungsten Cluster (Cluster-Extractor), follow the steps outlined here to ensure the cluster is configured correctly: Section 3.4.1, “Prepare: Replicating Data Out of a Cluster”

2. The applier can now be configured. Unpack the Tungsten Replicator distribution in staging directory:

```
shell> tar zxf tungsten-replicator-7.1.2-81.tar.gz
```

3. Change into the staging directory:

```
shell> cd tungsten-replicator-7.1.2-81
```

4. Configure the installation using tpm:

   Show Staging

   Show INI

```
shell> ./tools/tpm configure defaults \
    --reset \
    --user=tungsten \
    --install-directory=/opt/continuent \
    --profile-script=~/.bash_profile \
    --rest-api-admin-user=apiuser \
    --rest-api-admin-pass=secret

shell> ./tools/tpm configure alpha \
    --topology=master-slave \
    --master=sourcehost \
    --members=localhost \
    --datasource-type=redshift \
    --replication-host=redshift.us-east-1.redshift.amazonaws.com \
    --replication-user=awsRedshiftUser \
    --replication-password=awsRedshiftPass \
    --redshift-dbname=dev \
    --batch-enabled=true \
    --batch-load-template=redshift \
    --svc-applier-filters=dropstatementdata \
    --svc-applier-block-commit-interval=30s \
    --svc-applier-block-commit-size=250000
```

```
shell> vi /etc/tungsten/tungsten.ini
```

```
[defaults]
user=tungsten
install-directory=/opt/continuent
profile-script=~/.bash_profile
rest-api-admin-user=apiuser
rest-api-admin-pass=secret

[alpha]
topology=master-slave
master=sourcehost
members=localhost
datasource-type=redshift
replication-host=redshift.us-east-1.redshift.amazonaws.com
replication-user=awsRedshiftUser
replication-password=awsRedshiftPass
redshift-dbname=dev
batch-enabled=true
batch-load-template=redshift
svc-applier-filters=dropstatementdata
svc-applier-block-commit-interval=30s
svc-applier-block-commit-size=250000
```

   Configuration group `defaults`

   The description of each of the options is shown below; click the icon to hide this detail:

   Click the icon to show a detailed description of each argument.

   - `--reset` [422]

     `reset` [422]

     For staging configurations, deletes all pre-existing configuration information between updating with the new configuration values.

   - `--user=tungsten` [429]

     `user=tungsten` [429]

     System User

   - `--install-directory=/opt/continuent` [409]

`install-directory=/opt/continuent` [409]

Path to the directory where the active deployment will be installed. The configured directory will contain the software, THL and relay log information unless configured otherwise.

- `--profile-script=~/.bash_profile` [417]

  `profile-script=~/.bash_profile` [417]

  Append commands to include env.sh in this profile script

- `--rest-api-admin-user=apiuser` [422]

  `rest-api-admin-user=apiuser` [422]
  Optional: Must be specified along with rest-api-admin-pass if you wish to access the full API features and use the Dashboard GUI for cluster installations.

- `--rest-api-admin-pass=secret` [422]

  `rest-api-admin-pass=secret` [422]
  Optional: Must be specified along with rest-api-admin-user if you wish to access the full API features.

Configuration group **`alpha`**

The description of each of the options is shown below; click the icon to hide this detail:

Click the icon to show a detailed description of each argument.

- `--topology=master-slave` [429]

  `topology=master-slave` [429]

  Replication topology for the dataservice.

- `--master=sourcehost` [412]

  `master=sourcehost` [412]

  The hostname of the primary (extractor) within the current service.

- `--members=localhost` [413]

  `members=localhost` [413]

  Hostnames for the dataservice members

- `--datasource-type=redshift` [402]

  `datasource-type=redshift` [402]

  Database type

- `--replication-host=redshift.us-east-1.redshift.amazonaws.com` [420]

  `replication-host=redshift.us-east-1.redshift.amazonaws.com` [420]

  Hostname of the datasource where the database is located. If the specified hostname matches the current host or member name, the database is assumed to be local. If the hostnames do not match, extraction is assumed to be via remote access. For MySQL hosts, this configures a remote replication Replica (relay) connection.

- `--replication-user=awsRedshiftUser` [421]

  `replication-user=awsRedshiftUser` [421]

  For databases that required authentication, the username to use when connecting to the database using the corresponding connection method (native, JDBC, etc.).

- `--replication-password=awsRedshiftPass` [420]

  `replication-password=awsRedshiftPass` [420]

  The password to be used when connecting to the database using the corresponding `--replication-user` [421].

- `--redshift-dbname=dev` [418]

  `redshift-dbname=dev` [418]

  Name of the Redshift database to replicate into

- `--batch-enabled=true` [397]

  `batch-enabled=true` [397]

  Should the replicator service use a batch applier

- `--batch-load-template=redshift` [398]

  `batch-load-template=redshift` [398]

  Value for the loadBatchTemplate property

- `--svc-applier-filters=dropstatementdata` [424]

  `svc-applier-filters=dropstatementdata` [424]

  Replication service applier filters

- `--svc-applier-block-commit-interval=30s` [424]

  `svc-applier-block-commit-interval=30s` [424]

  Minimum interval between commits

- `--svc-applier-block-commit-size=250000` [424]

  `svc-applier-block-commit-size=250000` [424]

  Applier block commit size (min 1)

5. If your MySQL source is a Tungsten Cluster, ensure the additional steps below are also included in your applier configuration

   First, prepare the required filter configuration file as follows on the Redshift applier host(s) only:

```
shell> mkdir -p /opt/continuent/share/
shell> cp tungsten-replicator/support/filters-config/convertstringfrommysql.json /opt/continuent/share/
```

   Then, include the following parameters in the configuration

```
property=replicator.stage.remote-to-thl.filters=convertstringfrommysql
property=replicator.filter.convertstringfrommysql.definitionsFile=/opt/continuent/share/convertstringfrommysql.json
```

6. 

   **Note**

   If you plan to make full use of the REST API (which is enabled by default) you will need to also configure a username and password for API access. This must be done by specifying the following options in your configuration:

```
rest-api-admin-user=tungsten
rest-api-admin-pass=secret
```

7. Once the prerequisites and configuring of the installation has been completed, the software can be installed:

```
shell> ./tools/tpm install
```

If the installation process fails, check the output of the `/tmp/tungsten-configure.log` file for more information about the root cause.

On the host that is loading data into Redshift, create the `s3-config-servicename.json` file and then copy that file into the `share` directory within the installed directory on that host. For example:

```
shell> cp s3-config-servicename.json /opt/continuent/share/
```

Now the services can be started:

```
shell> replicator start
```

Once the service is configured and running, the service can be monitored as normal using the trepctl command. See Section 4.2.6, “Management and Monitoring of Amazon Redshift Deployments” for more information.

## 4.2.4. Verifying your Redshift Installation

1. Create a database within your source MySQL instance:

```
mysql> CREATE DATABASE redtest;
```

2. Create a table within your source MySQL instance:

```
mysql> CREATE TABLE redtest.msg (id INT PRIMARY KEY AUTO_INCREMENT,msg CHAR(80));
```

3. Create a schema for the tables:

```
redshift> CREATE SCHEMA redtest;
```

4. Create a staging table within your Redshift instance:

```
redshift> CREATE TABLE redtest.stage_xxx_msg (tungsten_opcode CHAR(1), \
    tungsten_seqno INT, tungsten_row_id INT,tungsten_date CHAR(30),id INT,msg CHAR(80));
```

5. Create the target table:

```
redshift> CREATE TABLE redtest.msg (id INT,msg CHAR(80));
```

6. Insert some data within your MySQL source instance:

```
mysql> INSERT INTO redtest.msg VALUES (0,'First');
Query OK, 1 row affected (0.04 sec)

mysql> INSERT INTO redtest.msg VALUES (0,'Second');
Query OK, 1 row affected (0.04 sec)

mysql> INSERT INTO redtest.msg VALUES (0,'Third');
Query OK, 1 row affected (0.04 sec)

mysql> UPDATE redtest.msg SET msg = 'This is the first update of the second row' WHERE ID = 2;
```

7. Check the replicator status on the applier (`host2`):

```
shell> trepctl status
```

   There should be 5 transactions replicated.

8. Check the table within Redshift:

```
redshift> SELECT * FROM redtest.msg;
1 First
3 Third
2 This is the first update of the second row
```

## 4.2.5. Keeping CDC Information

The Redshift applier can keep the CDC data, that is, the raw CDC CSV data that is recorded and replicated during the loading process, rather than simply cleaning up the CDC files and deleting them. The CDC data can be useful if you want to be able to monitor data changes over time.

The process works as follows:

1. Batch applier generates CSV files.
2. Batch applier loads the CSV data into the staging tables.
3. Batch applier loads the CSV data into the CDC tables.
4. Staging data is merged with the base table data.
5. Staging data is deleted.

Unlike the staging and base table information, the data in the CDC tables is kept forever, without removing any of the processed information. Using this data you can report on change information over time for different data sets, or even recreate datasets at a specific time by using the change information.

To enable this feature:

1. When creating the DDL for the staging and base tables, also create the table information for the CDC data for each table. The actual format of the information is the same as the staging table data, and can be created using ddlscan:

```
shell> ddlscan -service my_red -db test \
    -template ddl-mysql-redshift-staging.vm \
    -opt renameSchema cdc_{schema} -opt renameTable {table}_cdc
```

2. In the configuration file, `s3-config-svc.json` for each service, specify the name of the table to be used when storing the CDC information using the `storeCDCIn` field. This should specify the table template to be used, with the schema and table name being automatically replaced by the load script. The structure should match the structure used by ddlscan to define the CDC tables:

```
{
  "awsS3Path" : "s3://your-bucket-for-redshift/redshift-test",
  "awsAccessKey" : "access-key-id",
  "awsSecretKey" : "secret-access-key",
  "storeCDCIn" : "cdc_{schema}.{table}_cdc"
}
```

3. Restart the replicator using replicator restart to update the configuration.

## 4.2.6. Management and Monitoring of Amazon Redshift Deployments

Monitoring a Amazon Redshift replication scenario requires checking the status of both the Extractor - extracting data from MySQL - and the Applier which retrieves the remote THL information and applies it to Amazon Redshift.

```
shell> trepctl status
Processing status command...
NAME                     VALUE
----                     -----
appliedLastEventId     : mysql-bin.000006:0000000000002857;-1
appliedLastSeqno       : 15
appliedLatency         : 1.918
autoRecoveryEnabled    : false
autoRecoveryTotal      : 0
channels               : 1
clusterName            : alpha
currentEventId         : mysql-bin.000006:0000000000002857
currentTimeMillis      : 1407336195165
dataServerHost         : redshift1
extensions             :
host                   : redshift1
latestEpochNumber      : 8
masterConnectUri       : thl://localhost:/
masterListenUri        : thl://redshift1:2112/
maximumStoredSeqNo     : 15
minimumStoredSeqNo     : 0
offlineRequests        : NONE
pendingError           : NONE
pendingErrorCode       : NONE
pendingErrorEventId    : NONE
pendingErrorSeqno      : -1
pendingExceptionMessage: NONE
pipelineSource         : jdbc:mysql:thin://redshift1:3306/tungsten_alpha
relativeLatency        : 35.164
resourcePrecedence     : 99
rmiPort                : 10000
role                   : master
seqnoType              : java.lang.Long
serviceName            : alpha
serviceType            : local
simpleServiceName      : alpha
siteName               : default
sourceId               : redshift1
state                  : ONLINE
timeInStateSeconds     : 34.807
transitioningTo        :
uptimeSeconds          : 36.493
useSSLConnection       : false
version                : Tungsten Replicator 7.1.2 build 81
Finished status command...
```

On the Applier, the output of trepctl shows the current sequence number and applier status:

```
shell> trepctl status
Processing status command...
NAME                     VALUE
----                     -----
appliedLastEventId     : mysql-bin.000006:0000000000002857;-1
appliedLastSeqno       : 15
appliedLatency         : 154.748
```

```
autoRecoveryEnabled    : false
autoRecoveryTotal      : 0
channels               : 1
clusterName            : alpha
currentEventId         : NONE
currentTimeMillis      : 1407336316454
dataServerHost         : redshift.us-east-1.redshift.amazonaws.com
extensions             :
host                   : redshift.us-east-1.redshift.amazonaws.com
latestEpochNumber      : 8
masterConnectUri       : thl://redshift1:2112/
masterListenUri        : null
maximumStoredSeqNo     : 15
minimumStoredSeqNo     : 0
offlineRequests        : NONE
pendingError           : NONE
pendingErrorCode       : NONE
pendingErrorEventId    : NONE
pendingErrorSeqno      : -1
pendingExceptionMessage: NONE
pipelineSource         : thl://redshift1:2112/
relativeLatency        : 156.454
resourcePrecedence     : 99
rmiPort                : 10000
role                   : slave
seqnoType              : java.lang.Long
serviceName            : alpha
serviceType            : local
simpleServiceName      : alpha
siteName               : default
sourceId               : redshift.us-east-1.redshift.amazonaws.com
state                  : ONLINE
timeInStateSeconds     : 2.28
transitioningTo        :
uptimeSeconds          : 524104.751
useSSLConnection       : false
version                : Tungsten Replicator 7.1.2 build 81
Finished status command...
```

The *appliedLastSeqno* should match as normal. Because of the batching of transactions the *appliedLatency* may be much higher than a normal MySQL to MySQL replication.

The batch loading parameters controlling the batching of data can be tuned and update by studying the output from the `trepsvc.log` log file. The log will show a line containing the number of rows updated:

```
INFO  scripting.JavascriptExecutor COUNT: 4
```

See Section 12.1, “Block Commit” for more information on these parameters.

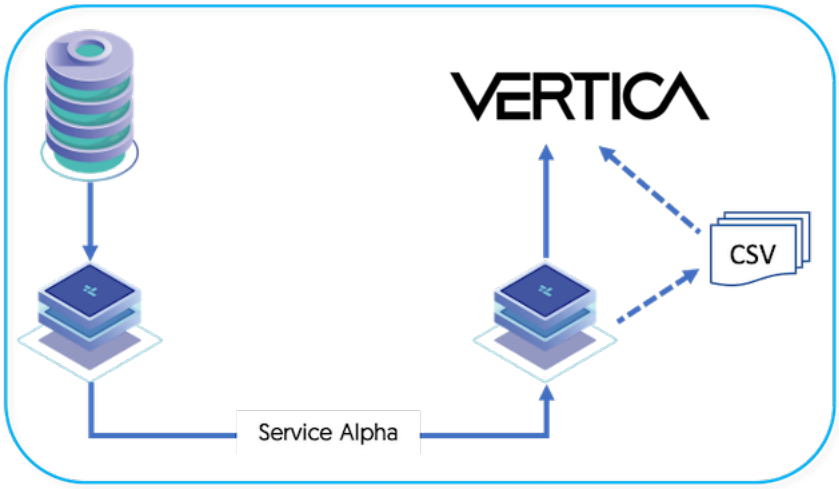
# 4.3. Deploying the Vertica Applier

Hewlett-Packard's Vertica provides support for BigData, SQL-based analysis and processing. Integration with MySQL enables data to be replicated live from the MySQL database directly into Vertica without the need to manually export and import the data.

Replication to Vertica operates as follows:

- Data is extracted from the source database into THL.
- When extracting the data from the THL, the Vertica replicator writes the data into CSV files according to the name of the source tables. The files contain all of the row-based data, including the global transaction ID generated by Tungsten Replicator during replication, and the operation type (insert, delete, etc) as part of the CSV data.
- The CSV data is then loaded into Vertica into staging tables.
- SQL statements are then executed to perform updates on the live version of the tables, using the CSV, batch loaded, information, deleting old rows, and inserting the new data when performing updates to work effectively within the confines of Vertica operation.

Figure 4.4. Topologies: Replicating to Vertica




Setting up replication requires setting up both the Extractor and Applier components as two different configurations, one for MySQL and the other for Vertica. Replication also requires some additional steps to ensure that the Vertica host is ready to accept the replicated data that has been extracted. Tungsten Replicator uses all the tools required to perform these operations during the installation and setup.

## 4.3.1. Preparing for Vertica Deployments

Preparing the hosts for the replication process requires setting some key configuration parameters within the MySQL server to ensure that data is stored and written correctly. On the Vertica side, the database and schema must be created using the existing schema definition so that the databases and tables exist within Vertica.

Source Host

Configure the source and target hosts following the prerequisites outlined in Appendix B, *Prerequisites* then follow the appropriate steps for the required extractor topology outlined in Chapter 3, *Deploying MySQL Extractors*.

Vertica Host

On the Vertica host, you need to perform some preparation of the destination database, first creating the database, and then creating the tables that are to be replicated.

- Create a database (if you want to use a different one than those already configured), and a schema that will contain the Tungsten data about the current replication position:

```
shell> vsql -Udbadmin -wsecret bigdata
Welcome to vsql, the Vertica Analytic Database v5.1.1-0 interactive terminal.

Type:  \h for help with SQL commands
       \? for help with vsql commands
       \g or terminate with semicolon to execute query
       \q to quit

bigdata=> create schema tungsten_alpha;
```

  The schema will be used only by Tungsten Replicator to store metadata about the replication process.

- Locate the Vertica JDBC driver. This can be downloaded separately from the Vertica website. The driver will need to be copied into the Tungsten Replicator `lib` directory.

```
shell> cp vertica-jdbc-7.1.2-0.jar tungsten-replicator-7.1.2-81/tungsten-replicator/lib/
```

- You need to create tables within Vertica according to the databases and tables that need to be replicated; the tables are not automatically created for you. From a Tungsten Replicator deployment directory, the ddlscan command can be used to identify the existing tables, and create table definitions for use within Vertica.

  To use ddlscan, the template for Vertica must be specified, along with the user/password information to connect to the source database to collect the schema definitions. The tool should be run from the templates directory.

  The tool will need to be executed twice, the first time generates the live table definitions:

```
shell> cd tungsten-replicator-7.1.2-81
shell> cd tungsten-replicator/samples/extensions/velocity/
shell> ddlscan -user tungsten -url 'jdbc:mysql:thin://host1:13306/access_log' -pass password \
    -template ddl-mysql-vertica.vm -db access_log
/*
SQL generated on Fri Sep 06 14:37:40 BST 2013 by ./ddlscan utility of Tungsten

url = jdbc:mysql:thin://host1:13306/access_log
user = tungsten
dbName = access_log
*/
CREATE SCHEMA access_log;

DROP TABLE access_log.access_log;

CREATE TABLE access_log.access_log
(
  id INT ,
  userid INT ,
  datetime INT ,
  session CHAR(30) ,
  operation CHAR(80) ,
  opdata CHAR(80)  ) ORDER BY id;
...
```

The output should be redirected to a file and then used to create tables within Vertica:

```
shell> ddlscan -user tungsten -url 'jdbc:mysql:thin://host1:13306/access_log' -pass password \
    -template ddl-mysql-vertica.vm -db access_log >access_log.ddl
```

The output of the command should be checked to ensure that the table definitions are correct.

The file can then be applied to Vertica:

```
shell> cat access_log.ddl | vsql -Udbadmin -wsecret bigdata
```

This generates the table definitions for live data. The process should be repeated to create the table definitions for the staging data by using te staging template:

```
shell> ddlscan -user tungsten -url 'jdbc:mysql:thin://host1:13306/access_log' -pass password \
    -template ddl-mysql-vertica-staging.vm -db access_log >access_log.ddl-staging
```

Then applied to Vertica:

```
shell> cat access_log.ddl-staging | vsql -Udbadmin -wsecret bigdata
```

The process should be repeated for each database that will be replicated.

Once the preparation of the MySQL and Vertica databases are ready, you can proceed to installing Tungsten Replicator

## 4.3.2. Install Vertica Applier

1. Before installing the applier, the following additions need adding to the extractor configuration. Apply the following parameter to the extractor configuration before installing the applier

   Add the following the `/etc/tungsten/tungsten.ini`

   ```
   [alpha]
   ...Existing Replicator Config...
   enable-heterogeneous-service=true

   shell> tpm update
   ```

   **Note**

   > The above step is only applicable for standalone extractors. If you are configuring replications from an existing Tungsten Cluster (Cluster-Extractor), follow the steps outlined here to ensure the cluster is configured correctly: Section 3.4.1, “Prepare: Replicating Data Out of a Cluster”

2. The applier can now be configured.

   Unpack the Tungsten Replicator distribution in staging directory:

   ```
   shell> tar zxf tungsten-replicator-7.1.2-81.tar.gz
   ```

3. Change into the staging directory:

```
shell> cd tungsten-replicator-7.1.2-81
```

4. Locate the Vertica JDBC driver. This can be downloaded separately from the Vertica website. The driver will need to be copied into the Tungsten Replicator `lib` directory.

```
shell> cp vertica-jdbc-7.1.2-0.jar tungsten-replicator-7.1.2-81/tungsten-replicator/lib/
```

5. Configure the installation using tpm:

   Show Staging

   Show INI

```
shell> ./tools/tpm configure defaults \
    --reset \
    --user=tungsten \
    --install-directory=/opt/continuent \
    --profile-script=~/.bash_profile \
    --skip-validation-check=HostsFileCheck \
    --skip-validation-check=InstallerMasterSlaveCheck \
    --rest-api-admin-user=apiuser \
    --rest-api-admin-pass=secret

shell> ./tools/tpm configure alpha \
    --topology=master-slave \
    --master=sourcehost \
    --members=localhost \
    --datasource-type=vertica \
    --replication-user=dbadmin \
    --replication-password=password \
    --vertica-dbname=dev \
    --batch-enabled=true \
    --batch-load-template=vertica6 \
    --batch-load-language=js \
    --replication-port=5433 \
    --svc-applier-filters=dropstatementdata \
    --svc-applier-block-commit-interval=30s \
    --svc-applier-block-commit-size=25000 \
    --disable-relay-logs=true
```

```
shell> vi /etc/tungsten/tungsten.ini
```

```
[defaults]
user=tungsten
install-directory=/opt/continuent
profile-script=~/.bash_profile
skip-validation-check=HostsFileCheck
skip-validation-check=InstallerMasterSlaveCheck
rest-api-admin-user=apiuser
rest-api-admin-pass=secret

[alpha]
topology=master-slave
master=sourcehost
members=localhost
datasource-type=vertica
replication-user=dbadmin
replication-password=password
vertica-dbname=dev
batch-enabled=true
batch-load-template=vertica6
batch-load-language=js
replication-port=5433
svc-applier-filters=dropstatementdata
svc-applier-block-commit-interval=30s
svc-applier-block-commit-size=25000
disable-relay-logs=true
```

   Configuration group **`defaults`**

   The description of each of the options is shown below; click the icon to hide this detail:

   Click the icon to show a detailed description of each argument.

   - `--reset` [422]

     `reset` [422]

     For staging configurations, deletes all pre-existing configuration information between updating with the new configuration values.

- `--user=tungsten` [429]

  `user=tungsten` [429]

  System User

- `--install-directory=/opt/continuent` [409]

  `install-directory=/opt/continuent` [409]

  Path to the directory where the active deployment will be installed. The configured directory will contain the software, THL and relay log information unless configured otherwise.

- `--profile-script=~/.bash_profile` [417]

  `profile-script=~/.bash_profile` [417]

  Append commands to include env.sh in this profile script

- `--skip-validation-check=HostsFileCheck` [369]

  `skip-validation-check=HostsFileCheck` [369]

  The `--skip-validation-check` [369] disables a given validation check. If any validation check fails, the installation, validation or configuration will automatically stop.

  > **Warning**
  >
  > Using this option enables you to bypass the specified check, although skipping a check may lead to an invalid or non-working configuration.

  You can identify a given check if an error or warning has been raised during configuration. For example, the default table type check:

  ```
  ...
  ERROR >> centos >> The datasource root@centos:3306 (WITH PASSWORD) »
   uses MyISAM as the default storage engine (MySQLDefaultTableTypeCheck)
  ...
  ```

  The check in this case is `MySQLDefaultTableTypeCheck` [380], and could be ignored using `--skip-validation-check=MySQLDefaultTableTypeCheck` [369].

  Setting both `--skip-validation-check` [369] and `--enable-validation-check` [367] is equivalent to explicitly disabling the specified check.

- `--skip-validation-check=InstallerMasterSlaveCheck` [369]

  `skip-validation-check=InstallerMasterSlaveCheck` [369]

  The `--skip-validation-check` [369] disables a given validation check. If any validation check fails, the installation, validation or configuration will automatically stop.

  > **Warning**
  >
  > Using this option enables you to bypass the specified check, although skipping a check may lead to an invalid or non-working configuration.

  You can identify a given check if an error or warning has been raised during configuration. For example, the default table type check:

  ```
  ...
  ERROR >> centos >> The datasource root@centos:3306 (WITH PASSWORD) »
   uses MyISAM as the default storage engine (MySQLDefaultTableTypeCheck)
  ...
  ```

  The check in this case is `MySQLDefaultTableTypeCheck` [380], and could be ignored using `--skip-validation-check=MySQLDefaultTableTypeCheck` [369].

  Setting both `--skip-validation-check` [369] and `--enable-validation-check` [367] is equivalent to explicitly disabling the specified check.

- `--rest-api-admin-user=apiuser` [422]

  `rest-api-admin-user=apiuser` [422]

  Optional: Must be specified along with rest-api-admin-pass if you wish to access the full API features and use the Dashboard GUI for cluster installations.

- `--rest-api-admin-pass=secret` [422]

  `rest-api-admin-pass=secret` [422]
  Optional: Must be specified along with rest-api-admin-user if you wish to access the full API features.

Configuration group **`alpha`**

The description of each of the options is shown below; click the icon to hide this detail:

Click the icon to show a detailed description of each argument.

- `--topology=master-slave` [429]

  `topology=master-slave` [429]

  Replication topology for the dataservice.

- `--master=sourcehost` [412]

  `master=sourcehost` [412]

  The hostname of the primary (extractor) within the current service.

- `--members=localhost` [413]

  `members=localhost` [413]

  Hostnames for the dataservice members

- `--datasource-type=vertica` [402]

  `datasource-type=vertica` [402]

  Database type

- `--replication-user=dbadmin` [421]

  `replication-user=dbadmin` [421]

  For databases that required authentication, the username to use when connecting to the database using the corresponding connection method (native, JDBC, etc.).

- `--replication-password=password` [420]

  `replication-password=password` [420]

  The password to be used when connecting to the database using the corresponding `--replication-user` [421].

- `--vertica-dbname=dev` [430]

  `vertica-dbname=dev` [430]

  Name of the database to replicate into

- `--batch-enabled=true` [397]

  `batch-enabled=true` [397]

  Should the replicator service use a batch applier

- `--batch-load-template=vertica6` [398]

  `batch-load-template=vertica6` [398]

  Value for the loadBatchTemplate property

- `--batch-load-language=js` [398]

  `batch-load-language=js` [398]

  Which script language to use for batch loading

- `--replication-port=5433` [421]

  `replication-port=5433` [421]

  The network port used to connect to the database server. The default port used depends on the database being configured.

- `--svc-applier-filters=dropstatementdata` [424]

  `svc-applier-filters=dropstatementdata` [424]

  Replication service applier filters

- `--svc-applier-block-commit-interval=30s` [424]

  `svc-applier-block-commit-interval=30s` [424]

  Minimum interval between commits

- `--svc-applier-block-commit-size=25000` [424]

  `svc-applier-block-commit-size=25000` [424]

  Applier block commit size (min 1)

- `--disable-relay-logs=true` [404]

  `disable-relay-logs=true` [404]

  Disable the use of relay-logs?

6. **Note**

   If you plan to make full use of the REST API (which is enabled by default) you will need to also configure a username and password for API access. This must be done by specifying the following options in your configuration:

   ```
   rest-api-admin-user=tungsten
   rest-api-admin-pass=secret
   ```

7. Once the prerequisites and configuring of the installation has been completed, the software can be installed:

   ```
   shell> ./tools/tpm install
   ```

If you encounter problems during the installation, check the output of the `/tmp/tungsten-configure.log` file for more information about the root cause.

Once the service is configured and running, the service can be monitored as normal using the trepctl command. See Section 4.3.3, “Management and Monitoring of Vertica Deployments” for more information.

## 4.3.3. Management and Monitoring of Vertica Deployments

Monitoring a Vertica replication scenario requires checking the status of both the Extractor - extracting data from MySQL - and the Applier which retrieves the remote THL information and applies it to Vertica.

```
shell> trepctl status
Processing status command...
NAME                     VALUE
----                     -----
appliedLastEventId     : mysql-bin.000012:0000000128889042;0
appliedLastSeqno       : 1070
appliedLatency         : 22.537
channels               : 1
clusterName            : alpha
currentEventId         : mysql-bin.000012:0000000128889042
currentTimeMillis      : 1378489888477
dataServerHost         : mysqldb01
extensions             :
latestEpochNumber      : 897
masterConnectUri       : thl://localhost:/
masterListenUri        : thl://mysqldb01:2112/
maximumStoredSeqNo     : 1070
minimumStoredSeqNo     : 0
offlineRequests        : NONE
pendingError           : NONE
```

```
pendingErrorCode       : NONE
pendingErrorEventId    : NONE
pendingErrorSeqno      : -1
pendingExceptionMessage: NONE
pipelineSource         : jdbc:mysql:thin://mysqldb01:13306/
relativeLatency        : 691980.477
resourcePrecedence     : 99
rmiPort                : 10000
role                   : master
seqnoType              : java.lang.Long
serviceName            : alpha
serviceType            : local
simpleServiceName      : alpha
siteName               : default
sourceId               : mysqldb01
state                  : ONLINE
timeInStateSeconds     : 694039.058
transitioningTo        :
uptimeSeconds          : 694041.81
useSSLConnection       : false
version                : Tungsten Replicator 7.1.2 build 81
Finished status command...
```

On the Applier, the output of trepctl shows the current sequence number and applier status:

```
shell> trepctl status
Processing status command...
NAME                     VALUE
----                     -----
appliedLastEventId     : mysql-bin.000012:0000000128889042;0
appliedLastSeqno       : 1070
appliedLatency         : 78.302
channels               : 1
clusterName            : default
currentEventId         : NONE
currentTimeMillis      : 1378479271609
dataServerHost         : vertica01
extensions             :
latestEpochNumber      : 897
masterConnectUri       : thl://mysqldb01:2112/
masterListenUri        : null
maximumStoredSeqNo     : 1070
minimumStoredSeqNo     : 0
offlineRequests        : NONE
pendingError           : NONE
pendingErrorCode       : NONE
pendingErrorEventId    : NONE
pendingErrorSeqno      : -1
pendingExceptionMessage: NONE
pipelineSource         : thl://mysqldb01:2112/
relativeLatency        : 681363.609
resourcePrecedence     : 99
rmiPort                : 10000
role                   : slave
seqnoType              : java.lang.Long
serviceName            : alpha
serviceType            : local
simpleServiceName      : alpha
siteName               : default
sourceId               : vertica01
state                  : ONLINE
timeInStateSeconds     : 681486.806
transitioningTo        :
uptimeSeconds          : 689922.693
useSSLConnection       : false
version                : Tungsten Replicator 7.1.2 build 81
Finished status command...
```

The *appliedLastSeqno* should match as normal. Because of the batching of transactions the *appliedLatency* may be much higher than a normal MySQL to MySQL replication.

## 4.3.4. Troubleshooting Vertica Installations

The following items detail some of the more common problems with replication through to Vertica. Often the underlying issue is related to the data types, the data format, or the number of columns.

- If the following is reported by the replicator:

```
pendingError           : Replicator unable to go online due to error »
  Operation failed: Online operation failed (Unable to prepare plugin: class »
  name=com.continuent.tungsten.replicator.datasource.DataSourceService »
```

```
  message=[Unable to load driver: com.vertica.jdbc.Driver])
state                  : OFFLINE:ERROR
```

The Vertica JDBC driver is missing from the installation. The Vertica JDBC JAR file must have been placed into the `tungsten-replicator/lib` directory within the release diectory before running tpm update or tpm install.

- The following error:

```
pendingExceptionMessage: Invalid write to CSV file: name=/opt/continuent/tmp/staging/alpha/staging0/test-msg-1.csv »
  table=test.msg table_columns=schemaname,schemahash csv_columns=tungsten_opcode,tungsten_seqno, »
  tungsten_row_id,tungsten_commit_timestamp,nullschemaname,schemahash
```

Indicates the source THL has been not been marked up correctly. Either the `colnames` filter has not been enabled, or the `--enable-batch-service` [405] has not been confifgred during installation. This means that the source THL is not being populated with the right information, either the full list of columns, or the column names and primary key information is incorrect. The configuration should be updated, and then the THL on both the Extractor and Applier should be recreated by using trepctl reset.

- If you get an error similar to the following:

```
pendingExceptionMessage: CSV loading failed: schema=test table=msg CSV »
 file=/opt/continuent/tmp/staging/alpha/staging0/test-msg-1.csv »
 message=com.continuent.tungsten.replicator.ReplicatorException: Incoming table data »
 has no primary keys: test.msg »
 (/opt/continuent/tungsten/tungsten-replicator/appliers/batch/vertica6.js#70)
```

Either the `pkey` filter has not been enabled, or the source tables on the source database do not contain primary keys. This means that the source THL is not being populated with the primary key information from the table which is requird in order to load into Vertica through the batch mechanism. The configuration should be updated, and then the THL on both the Extractor and Applier should be recreated by using trepctl reset.

- The following error indicates that the incoming data could not be loaded into the staging table within Vertica:

```
pendingError  : Stage task failed: q-to-dbms
pendingExceptionMessage: CSV loading failed: schema=blog table=article CSV »
  file=/tmp/staging/alpha/staging0/blog-article-432.csv »
  message=com.continuent.tungsten.replicator.ReplicatorException:
  LOAD DATA ROW count does not match: sql=COPY blog.stage_xxx_article »
  FROM '/tmp/staging/alpha/staging0/blog-article-432.csv' »
  DIRECT NULL 'null' DELIMITER ',' ENCLOSED BY '"' »
  expected_copy_rows=3614 rows=2233 ; exceptions are in »
  /tmp/tungsten_vertica_blog.article.exceptions »
  (../../tungsten-replicator//samples/scripts/batch/vertica6.js#67)
```

There are a number of possible reasons for this. The actual reasons can be found in the exceptions file which is generated, the error message contains the location. In this example `/tmp/tungsten_vertica_blog.article.exceptions`. Possible reasons include:

  - Mismatch in the number of columns in the source file and the target table. Check the source and target tables match, including the four special fields used in all staging tables.
  - Mismatch in the data types of one or more of the columns in target table. Check the source and target table definitions match, or at least support the corresponding data. For example, the column size, length or format is correct. Loading character data into numeric columns, or floating point values into integer columns for example is not supported.
  - Badly formatted CSV file. This happens when the incoming data contains newliness or commas or other data that is incompatible with the CSV format. The CSV file should have been kept, the location is also in the error message. Examine the file and check the format. You may need to enable filters to modify and 'clean' the data so that it is more compatible with the CSV format.

- Remember that changes to the DDL within the source database are not automatically replicated to Vertica. Changes to the table definitions, additional tables, or additional databases, must all be updated manually within Vertica.

- If you get errors similar to:

```
stage_xxx_access_log does not exist
```

When loading into Vertica, it means that the staging tables have not created correctly. Check the steps for creating the staging tables using ddlscan in Section 4.3.1, “Preparing for Vertica Deployments”.

- Replication may fail if date types contain zero values, which are legal in MySQL. For example, the timestamp `0000-00-00 00:00:00` is valid in MySQL. An error reporting a mismatch in the values will be reported when applying the data into Vertica, for example:

```
ERROR 2631:  Column "time" is of type timestamp but expression is of type int
HINT:  You will need to rewrite or cast the expression
```

Or:

```
ERROR 2992:  Date/time field value out of range: "0"
```

```
HINT:  Perhaps you need a different "datestyle" setting
```

To address this error, use the `zerodate2null` filter, which translates zero-value dates into a valid NULL value. This can be enabled by adding the `zerodate2null` filter to the applier stage when configuring the service using tpm:

```
shell> ./tools/tpm update alpha --repl-svc-applier-filters=zerodate2null
```

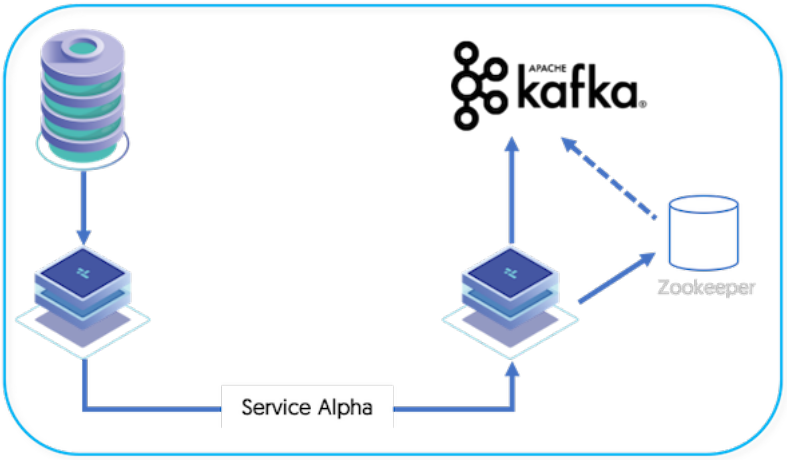
# 4.4. Deploying the Kafka Applier

Kafka is a highly scalable messaging platform that provides a method for distributing information through a series of messages organised by a specified topic. With Tungsten Replicator the incoming stream of data from the upstream replicator is converted, on a row by row basis, into a JSON document that contains the row information. A new message is created for each row, even from multiple-row transactions.

The deployment of Tungsten Replicator to Kafka service is slightly different. There are two parts to the process:

- Service Alpha on the Extractor, extracts the information from the MySQL binary log into THL.
- Service Alpha on the Applier, reads the information from the remote replicator as THL, and applies that to Kafka.

Figure 4.5. Topologies: Replicating to Kafka




With the Kafka applier, information is extracted from the source database using the row-format, column names and primary keys are identified, and translated to a JSON format, and then embedded into a larger Kafka message. The topic used is either composed from the schema name or can be configured to use an explicit topic type, and the generated information included in the Kafka message can include the source schema, table, and commit time information.

The transfer operates as follows:

1. Data is extracted from MySQL using the standard extractor, reading the row change data from the binlog.
2. The Section 11.4.5, “ColumnName Filter” filter is used to extract column name information from the database. This enables the row-change information to be tagged with the corresponding column information. The data changes, and corresponding row names, are stored in the THL.

   The Section 11.4.31, “PrimaryKey Filter” filter is used to add primary key information to row-based replication data.
3. The THL information is then applied to Kafka using the Kafka applier.

There are some additional considerations when applying to Kafka that should be taken into account:

- Because Kafka is a message queue and not a database, traditional transactional semantics are not supported. This means that although the data will be applied to Kafka as a message, there is no guarantee of transactional consistency. By default the applier will ensure that the message has been correctly received by the Kafka service, it is the responsibility of the Kafka environment and configuration to ensure delivery. The replicator.applier.dbms.zookeeperString can be used to ensure acknowledgements are received from the Kafka service.
- One message is sent for each row of source information in each transaction. For example, if 20 rows have been inserted or updated in a single transaction, then 20 separate Kafka messages will be generated.
- A separate message is broadcast for each operation, and includes the operation type. A single message will be broadcast for each row for each operation. So if 20 rows are delete, 20 messages are generated, each with the operation type.

- If replication fails in the middle of a large transaction, and the replicator goes `OFFLINE` [195], when the replicator goes online it may resend rows and messages.

The two replication services can operate on the same machine, (See Section 5.3, "Deploying Multiple Replicators on a Single Host") or they can be installed on two different machines.

## 4.4.1. Preparing for Kafka Replication

Configure the source and target hosts following the prerequisites outlined in Appendix B, *Prerequisites* then follow the appropriate steps for the required extractor topology outlined in Chapter 3, *Deploying MySQL Extractors*.

In general, it is easier to understand that a row within the MySQL table is converted into a single message on the Kafka side, the topic used is made up of the schema name and table name, and the message ID is composed of the primary key information, but can optionally include the schema and table name and primary key information.

For example, the following row within MySQL:

```
mysql> select * from messages where id = 99999 \G
*************************** 1. row ***************************
 id: 99999
 msg: Hello Kafka
 1 row in set (0.00 sec)
```

Is replicated into Kafka as a Kafka message using the topic `test_msg`:

```
{
   "_seqno" : "4865",
   "_source_table" : "msg",
   "_committime" : "2017-07-13 15:30:37.0",
   "_source_schema" : "test",
   "record" : {
      "msg" : "Hello Kafka",
      "id" : "2384726"
   },
   "_optype" : "INSERT"
}
```

In the output, the `record` contains the actualy record data, the other fields in the message are:

- _seqno — the THL sequence number of the transaction.
- _source_table — the source table. Inclusion of this information is optional.
- _committime — the original transaction commit time. Inclusion of this information is optional.
- _source_schema — the source schema. Inclusion of this information is optional.
- _optype — the operation type (INSERT, UPDATE, DELETE).

When preparing the hosts you must be aware of this translation of the different structures, as it will have an effect on the way the information is replicated from MySQL to Kafka.

MySQL Host

The data replicated from MySQL can be any data, although there are some known limitations and assumptions made on the way the information is transferred.

When configuring the extractor database and host, ensure heterogenous specific prerequisities have been included, see Section B.4.4, "MySQL Configuration for Heterogeneous Deployments"

For the best results when replicating, be aware of the following issues and limitations:

- Use primary keys on all tables. The use of primary keys will improve the lookup of information within Kafka when rows are updated. Without a primary key on a table a full table scan is performed, which can affect performance.
- MySQL `TEXT` columns are correctly replicated, but cannot be used as keys.
- MySQL `BLOB` columns are converted to text using the configured character type. Depending on the data that is being stored within the `BLOB`, the data may need to be custom converted. A filter can be written to convert and reformat the content as required.

Kafka Host

On the Kafka side, status information is stored into the Zookeeper instance used for configuring Kafka, and the Zookeeper and Kafka instances must be up and running before the replicator is first started. There are no specific configuration elements required on the Kafka host.

## 4.4.2. Install Kafka Applier

Installation of the Kafka replication requires special configuration of the Extractor and Applier hosts so that each is configured for the correct datasource type.

1. Before installing the applier, the following additions need adding to the extractor configuration. Apply the following parameter to the extractor configuration before installing the applier

   Add the following the `/etc/tungsten/tungsten.ini`

   ```
   [alpha]
   ...Existing Replicator Config...
   enable-heterogeneous-service=true

   shell> tpm update
   ```

   **Note**

   > The above step is only applicable for standalone extractors. If you are configuring replications from an existing Tungsten Cluster (Cluster-Extractor), follow the steps outlined here to ensure the cluster is configured correctly: Section 3.4.1, "Prepare: Replicating Data Out of a Cluster"

2. Unpack the Tungsten Replicator distribution in staging directory:

   ```
   shell> tar zxf tungsten-replicator-7.1.2-81.tar.gz
   ```

3. Change into the staging directory:

   ```
   shell> cd tungsten-replicator-7.1.2-81
   ```

4. Configure the installation using tpm:

   Show Staging

   Show INI

   ```
   shell> ./tools/tpm configure defaults \
       --reset \
       --install-directory=/opt/continuent \
       --profile-script=~/.bash_profile \
       --rest-api-admin-user=apiuser \
       --rest-api-admin-pass=secret

   shell> ./tools/tpm configure alpha \
       --master=sourcehost \
       --members=localhost \
       --datasource-type=kafka \
       --replication-user=root \
       --replication-password=null \
       --replication-port=9092 \
       --property=replicator.applier.dbms.zookeeperString=localhost:2181 \
       --property=replicator.applier.dbms.requireacks=1
   ```

   ```
   shell> vi /etc/tungsten/tungsten.ini
   ```

   ```
   [defaults]
   install-directory=/opt/continuent
   profile-script=~/.bash_profile
   rest-api-admin-user=apiuser
   rest-api-admin-pass=secret

   [alpha]
   master=sourcehost
   members=localhost
   datasource-type=kafka
   replication-user=root
   replication-password=null
   replication-port=9092
   property=replicator.applier.dbms.zookeeperString=localhost:2181
   property=replicator.applier.dbms.requireacks=1
   ```

   Configuration group **`defaults`**

   The description of each of the options is shown below; click the icon to hide this detail:

   Click the icon to show a detailed description of each argument.

   - `--reset` [422]

`reset` [422]

For staging configurations, deletes all pre-existing configuration information between updating with the new configuration values.

- `--install-directory=/opt/continuent` [409]

  `install-directory=/opt/continuent` [409]

  Path to the directory where the active deployment will be installed. The configured directory will contain the software, THL and relay log information unless configured otherwise.

- `--profile-script=~/.bash_profile` [417]

  `profile-script=~/.bash_profile` [417]

  Append commands to include env.sh in this profile script

- `--rest-api-admin-user=apiuser` [422]

  `rest-api-admin-user=apiuser` [422]
  Optional: Must be specified along with rest-api-admin-pass if you wish to access the full API features and use the Dashboard GUI for cluster installations.

- `--rest-api-admin-pass=secret` [422]

  `rest-api-admin-pass=secret` [422]
  Optional: Must be specified along with rest-api-admin-user if you wish to access the full API features.

Configuration group **`alpha`**

The description of each of the options is shown below; click the icon to hide this detail:

Click the icon to show a detailed description of each argument.

- `--master=sourcehost` [412]

  `master=sourcehost` [412]

  The hostname of the primary (extractor) within the current service.

- `--members=localhost` [413]

  `members=localhost` [413]

  Hostnames for the dataservice members

- `--datasource-type=kafka` [402]

  `datasource-type=kafka` [402]

  Database type

- `--replication-user=root` [421]

  `replication-user=root` [421]

  For databases that required authentication, the username to use when connecting to the database using the corresponding connection method (native, JDBC, etc.).

- `--replication-password=null` [420]

  `replication-password=null` [420]

  The password to be used when connecting to the database using the corresponding `--replication-user` [421].

- `--replication-port=9092` [421]

  `replication-port=9092` [421]

  The network port used to connect to the database server. The default port used depends on the database being configured.



5. If your MySQL source is a Tungsten Cluster, ensure the additional steps below are also included in your applier configuration

First, prepare the required filter configuration file as follows on the Kafka applier host(s) only:

```
shell> mkdir -p /opt/continuent/share/
shell> cp tungsten-replicator/support/filters-config/convertstringfrommysql.json /opt/continuent/share/
```

Then, include the following parameters in the configuration

```
property=replicator.stage.remote-to-thl.filters=convertstringfrommysql
property=replicator.filter.convertstringfrommysql.definitionsFile=/opt/continuent/share/convertstringfrommysql.json
```

6. **Note**

   If you plan to make full use of the REST API (which is enabled by default) you will need to also configure a username and password for API access. This must be done by specifying the following options in your configuration:

   ```
   rest-api-admin-user=tungsten
   rest-api-admin-pass=secret
   ```

7. Once the prerequisites and configuring of the installation has been completed, the software can be installed:

   ```
   shell> ./tools/tpm install
   ```

If you encounter problems during the installation, check the output of the `/tmp/tungsten-configure.log` file for more information about the root cause.

Once the service is configured and running, the service can be monitored as normal using the trepctl command. See Section 4.4.3, "Management and Monitoring of Kafka Deployments" for more information.

### 4.4.2.1. Optional Configuration Parameters for Kafka

A number of optional, configurable, properties are available that control how Tungsten Replicator applies and populates information when the data is written into Kafka. The following properties can by set during configuration using `--property=PROPERTYNAME=value` [368]:

Table 4.1. Optional Kafka Applier Properties

| Option | Description |
|---|---|
| replicator.applier.dbms.embedCommitTime | Sets whether the commit time for the source row is embedded into the document |
| replicator.applier.dbms.embedSchemaTable | Embed the source schema name and table name in the stored document |
| replicator.applier.dbms.enabletxinfo.kafka | Embeds transaction information (generated by the rowaddtxninfo filter) into each Kafka message |
| replicator.applier.dbms.enabletxninfoTopic | Embeds transaction information into a separate Kafka message broadcast on an independent channel from the one used by the actual database data. One message is sent per transaction or THL event. |
| replicator.applier.dbms.keyFormat | Determines the format of the message ID |
| replicator.applier.dbms.requireacks | Defines whether when writing messages to the Kafka cluster, how many acknowledgements from Kafka nodes is required |
| replicator.applier.dbms.retrycount | The number of retries for sending each message |
| replicator.applier.dbms.txninfoTopic | Sets the topic name for transaction messages |
| replicator.applier.dbms.zookeeperString | Connection string for Zookeeper, including hostname and port |

– `replicator.applier.dbms.embedCommitTime` [93]

| Option | `replicator.applier.dbms.embedCommitTime` [93] | |
|---|---|---|
| Description | Sets whether the commit time for the source row is embedded into the document | |
| Value Type | boolean | |
| Default | true | |
| Valid Values | false | Do not embed the source database commit time |
| | true | Embed the source database commit time into the stored document |

Embeds the commit time of the source database row into the document information:

```
{
 "_seqno" : "4865",
 "_source_table" : "msg",
 "_committime" : "2017-07-13 15:30:37.0",
 "_source_schema" : "test",
 "record" : {
 "msg" : "Hello Kafka",
 "id" : "2384726"
 },
 "_optype" : "INSERT"
}
```

– `replicator.applier.dbms.embedSchemaTable` [94]

| Option | `replicator.applier.dbms.embedSchemaTable` [94] | |
|---|---|---|
| Description | Embed the source schema name and table name in the stored document | |
| Value Type | boolean | |
| Default | true | |
| Valid Values | false | Do not embed the schema or database name in the document |
| | true | Embed the source schema name and database name into the stored document |

If enabled, the documented stored into Elasticsearch will include the source schema and database name. This can be used to identify the source of the information if the schema and table name is not being used for the index and type names (see replicator.applier.dbms.useSchemaAsIndex and replicator.applier.dbms.useTableAsType).

```
{
 "_seqno" : "4865",
 "_source_table" : "msg",
 "_committime" : "2017-07-13 15:30:37.0",
 "_source_schema" : "test",
 "record" : {
 "msg" : "Hello Kafka",
 "id" : "2384726"
 },
 "_optype" : "INSERT"
}
```

– `replicator.applier.dbms.enabletxinfo.kafka` [94]

| Option | `replicator.applier.dbms.enabletxinfo.kafka` [94] | |
|---|---|---|
| Description | Embeds transaction information (generated by the rowaddtxninfo filter) into each Kafka message | |
| Value Type | boolean | |
| Default | false | |
| Valid Values | false | Do not include transaction information in each |
| | true | Embed transaction information into each Kafka message |

Embeds information about the entire transaction information using the data provided by the `rowaddtxninfo` filter and other information embedded in each THL event into each message sent. The transaction information includes information about the entire transaction (row counts, event ID and tables modified) into each message. Since one message is normally sent for each row of data, by adding the information about the full transaction into the message it's possible to validate and identify what other messages may be part of a single transaction when the messages are being re-assembled by a Kafka client.

For example, when looking at a single message in Kafka, the message includes a `txninfo` section:

```
{
 "_source_table" : "msg",
 "_committime" : "2018-03-07 12:53:21.0",
 "record" : {
 "msg2" : "txinfo",
 "id" : "109",
 "msg" : "txinfo"
 },
 "_optype" : "INSERT",
 "_seqno" : "164",
 "txnInfo" : {
```

```
 "schema" : [
 {
 "schemaName" : "msg",
 "rowCount" : "1",
 "tableName" : "msg"
 },
 {
 "rowCount" : "2",
 "schemaName" : "msg",
 "tableName" : "msgsub"
 }
 ],
 "serviceName" : "alpha",
 "totalCount" : "3",
 "tungstenTransId" : "164",
 "firstRecordInTransaction" : "true"
 },
 "_source_schema" : "msg"
}
```

This block of the overall message includes the following objects and information:

- `schema`

  An array of the row counts within this transaction, with a row count included for each schema and table.

- `serviceName`

  The name of the Tungsten Replicator service that generated the message.

- `totalCount`

  The total number of rows modified within the entire transaction.

- `firstRecordInTransaction`

  If this field exists, it should always be set to true and indicats that this message was generated by the first row inserted, updated or deleted in the overall transaction. This effectively indicates the start of the overall transaction.

- `lastRecordInTransaction`

  If this field exists, it should always be set to true and indicats that this message was generated by the last row inserted, updated or deleted in the overall transaction. This effectively indicates the end of the overall transaction

Note that this information block is included in *every* message for each row within an overall transaction. The `firstRecordInTransaction` and `lastRecordInTransaction` can be used to identify the start and end of the transaction overall.

– `replicator.applier.dbms.enabletxninfoTopic` [95]

| Option | `replicator.applier.dbms.enabletxninfoTopic` [95] | |
|---|---|---|
| Description | Embeds transaction information into a separate Kafka message broadcast on an independent channel from the one used by the actual database data. One message is sent per transaction or THL event. | |
| Value Type | boolean | |
| Default | false | |
| Valid Values | false | Do not generate transaction information |
| | true | Send transaction information on a separate Kafka topic for each transaction |

If enabled, it sends a separate message on a Kafka topic containing information about the entire tranaction. The topic name can be configured by setting the replicator.applier.dbms.txninfoTopic property.

The default message sent will look like the following example:

```
{
 "txnInfo" : {
 "tungstenTransId" : "164",
 "schema" : [
 {
 "schemaName" : "msg",
 "rowCount" : "1",
 "tableName" : "msg"
 },
```

```
 {
 "schemaName" : "msg",
 "rowCount" : "2",
 "tableName" : "msgsub"
 }
 ],
 "totalCount" : "3",
 "serviceName" : "alpha"
 }
}
```

This block of the overall message includes the following objects and information:

- `schema`

  An array of the row counts within this transaction, with a row count included for each schema and table.

- `serviceName`

  The name of the Tungsten Replicator service that generated the message.

- `totalCount`

  The total number of rows modified within the entire transaction.

– `replicator.applier.dbms.keyFormat` [96]

| Option | `replicator.applier.dbms.keyFormat` [96] | |
|---|---|---|
| Description | Determines the format of the message ID | |
| Value Type | string | |
| Default | pkey | |
| Valid Values | pkey | Combine the primary key column values into a single string |
| | pkeyus | Combine the primary key column values into a single string joined by an underscore character |
| | tspkey | Combine the schema name, table name, and primary key column values into a single string joined by an underscore character |
| | tspkeyus | Combine the schema name, table name, and primary key column values into a single string |

Determines the format of the message ID used when sending the message into Kafka. For example, when configured to use `tspkeyus`, then the format of the message ID will consist of the schemaname, table name and primary key column information separated by underscores, `SCHEMANAME_TABLENAME_234`.

– `replicator.applier.dbms.requireacks` [96]

| Option | `replicator.applier.dbms.requireacks` [96] | |
|---|---|---|
| Description | Defines whether when writing messages to the Kafka cluster, how many acknowledgements from Kafka nodes is required | |
| Value Type | string | |
| Default | all | |
| Valid Values | 1 | Only the lead host should acknowledge receipt of the message |
| | all | All nodes should acknowledge receipt of the message |

Sets the acknowledgement counter for sending messages into the Kafka queue.

– `replicator.applier.dbms.retrycount` [96]

| Option | `replicator.applier.dbms.retrycount` [96] |
|---|---|
| Description | The number of retries for sending each message |
| Value Type | number |

| | |
|---|---|
| Default | 0 |

Determines the number of times the message will attempt to be sent before failure.

– `replicator.applier.dbms.txninfoTopic` [97]

| Option | `replicator.applier.dbms.txninfoTopic` [97] |
|---|---|
| Description | Sets the topic name for transaction messages |
| Value Type | string |
| Default | tungsten_transactions |

Sets the topic name to be used when sending independent transaction information messagesa about each THL event. See replicator.applier.dbms.addtxninfo.

– `replicator.applier.dbms.zookeeperString` [97]

| Option | `replicator.applier.dbms.zookeeperString` [97] |
|---|---|
| Description | Connection string for Zookeeper, including hostname and port |
| Value Type | string |
| Default | ${replicator.global.db.host}:2181 |

The string to be used when connecting to Zookeeper. The default is to use port 2181 on the host used by replicator.global.db.host.

## 4.4.3. Management and Monitoring of Kafka Deployments

Once the extractor and applier have been installed, services can be monitored using the trepctl command.

For example, to monitor the extractor status:

```
shell> trepctl status
appliedLastEventId     : mysql-bin.000009:0000000000002298;2340
appliedLastSeqno       : 10
appliedLatency         : 0.788
autoRecoveryEnabled    : false
autoRecoveryTotal      : 0
channels               : 1
clusterName            : alpha
currentEventId         : mysql-bin.000009:0000000000002298
currentTimeMillis      : 1498687871560
dataServerHost         : mysqlhost
extensions             :
host                   : mysqlhost
latestEpochNumber      : 0
masterConnectUri       : thl://localhost:/
masterListenUri        : thl://mysqlhost:2112/
maximumStoredSeqNo     : 10
minimumStoredSeqNo     : 0
offlineRequests        : NONE
pendingError           : NONE
pendingErrorCode       : NONE
pendingErrorEventId    : NONE
pendingErrorSeqno      : -1
pendingExceptionMessage: NONE
pipelineSource         : /var/lib/mysql
relativeLatency        : 99185.56
resourcePrecedence     : 99
rmiPort                : 10000
role                   : master
seqnoType              : java.lang.Long
serviceName            : east
serviceType            : local
simpleServiceName      : east
siteName               : default
sourceId               : mysqlhost
state                  : ONLINE
timeInStateSeconds     : 101347.786
timezone               : GMT
transitioningTo        :
uptimeSeconds          : 101358.88
useSSLConnection       : false
version                : Tungsten Replicator 7.1.2 build 81
Finished status command...
```

The replicator service operates just the same as a standard extractor service of a typical MySQL replication service.

The Kafka applier service can be accessed either remotely from the extractor:

```
shell> trepctl -host kafka status
...
```

Or locally on the Kafka host:

```
shell> trepctl status
Processing status command...
NAME                     VALUE
----                     -----
appliedLastEventId     : mysql-bin.000008:0000000000412301;0
appliedLastSeqno       : 1296
appliedLatency         : 10.253
channels               : 1
clusterName            : alpha
currentEventId         : NONE
currentTimeMillis      : 1377098139212
dataServerHost         : kafka
extensions             :
latestEpochNumber      : 1286
masterConnectUri       : thl://host1:2112/
masterListenUri        : null
maximumStoredSeqNo     : 1296
minimumStoredSeqNo     : 0
offlineRequests        : NONE
pendingError           : NONE
pendingErrorCode       : NONE
pendingErrorEventId    : NONE
pendingErrorSeqno      : -1
pendingExceptionMessage: NONE
pipelineSource         : thl://mysqlhost:2112/
relativeLatency        : 771.212
resourcePrecedence     : 99
rmiPort                : 10000
role                   : slave
seqnoType              : java.lang.Long
serviceName            : alpha
serviceType            : local
simpleServiceName      : alpha
siteName               : default
sourceId               : kafka
state                  : ONLINE
timeInStateSeconds     : 177783.343
transitioningTo        :
uptimeSeconds          : 180631.276
useSSLConnection       : false
version                : Tungsten Replicator 7.1.2 build 81
Finished status command...
```

Monitoring the status of replication between the source and target is also the same. The *appliedLastSeqno* still indicates the sequence number that has been applied to Kafka, and the event ID from Kafka can still be identified from *appliedLastEventId*.

Sequence numbers between the two hosts should match, as in a source/target deployment, but due to the method used to replicate, the applied latency may be higher.

To check for information within Kafka, use a tool or the kafka-console-consumer.sh command-line client:

```
shell> kafka-console-consumer.sh --topic test_msg --zookeeper localhost:2181
```

The output should be checked to ensure that information is being correctly replicated. If strings are shown as a hex value, for example:

```
"title" : "[B@7084a5c"
```

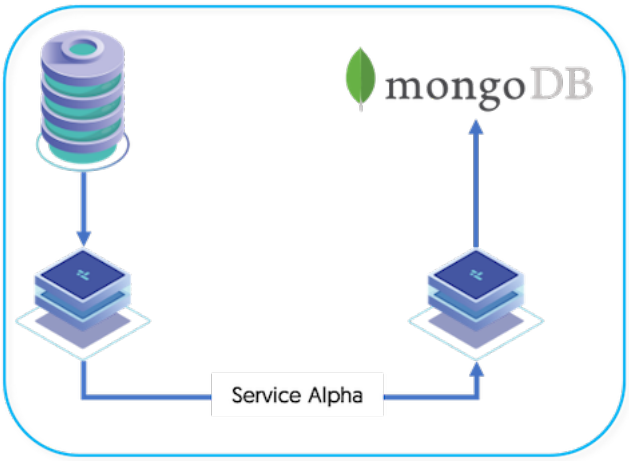
It probably indicates that UTF8 and/or `--mysql-use-bytes-for-string=false` [414] options were not used during installation. If you are reading from a cluster this is expected behavior, and you should enable the `convertstringfrommysql` filter as shown in the installation examples. In pure replicator scenarios, ensure that the `--mysql-use-bytes-for-string=false` [414] setting is enabled, or that you are using `--enable-heterogeneous-service` [406].

# 4.5. Deploying the MongoDB Applier

Deployment of a replication to MongoDB service is slightly different to other appliers, there are two parts to the process:

- Service Alpha on the Extractor, extracts the information from the MySQL binary log into THL.
- Service Alpha on the Applier reads the information from the remote replicator as THL, and applies that to MongoDB.

Figure 4.6. Topologies: Replicating to MongoDB




Basic reformatting and restructuring of the data is performed by translating the structure extracted from one database in row format and restructuring for application in a different format. A filter, the ColumnNameFilter, is used to extract the column names against the extracted row-based information.

With the MongoDB applier, information is extracted from the source database using the row-format, column names and primary keys are identified, and translated to the BSON (Binary JSON) format supported by MongoDB. The fields in the source row are converted to the key/value pairs within the generated BSON.

The transfer operates as follows:

1. Data is extracted from MySQL using the standard extractor, reading the row change data from the binlog.
2. The Section 11.4.5, "ColumnName Filter" filter is used to extract column name information from the database. This enables the row-change information to be tagged with the corresponding column information. The data changes, and corresponding row names, are stored in the THL.
3. The THL information is then applied to MongoDB using the MongoDB applier.

The two replication services can operate on the same machine, (See Section 5.3, "Deploying Multiple Replicators on a Single Host") or they can be installed on two different machines.

## 4.5.1. MongoDB Atlas Replication

The MongoDB applier can also be used to apply into a MongoDB Atlas instance.

The configuration for MongoDB Atlas is slightly different and follows a typical offboard applier process, similar in style to applying to Amazon Aurora Instances

Specific installation steps for MongoDB Atlas are outlined here Section 4.5.4, "Install MongoDB Atlas Applier"

## 4.5.2. Preparing for MongoDB Replication

Configure the source and target hosts following the prerequisites outlined in Appendix B, *Prerequisites* then follow the appropriate steps for the required extractor topology outlined in Chapter 3, *Deploying MySQL Extractors*.

During the replication process, data is exchanged from the MySQL database/table/row structure into corresponding MongoDB structures, as follows

| MySQL | MongoDB |
|---|---|
| Database | Database |
| Table | Collection |
| Row | Document |

In general, it is easier to understand that a row within the MySQL table is converted into a single document on the MongoDB side, and automatically added to a collection matching the table name.

For example, the following row within MySQL:

```
mysql> select * from recipe where recipeid = 1085 \G
```

```
*************************** 1. row ***************************
  recipeid: 1085
     title: Creamy egg and leek special
  subtitle:
  servings: 4
    active: 1
     parid: 0
    userid: 0
    rating: 0.0
 cumrating: 0.0
createdate: 0
1 row in set (0.00 sec)
```

Is replicated into the MongoDB document:

```
{
    "_id" : ObjectId("5212233584ae46ce07e427c3"),
    "recipeid" : "1085",
    "title" : "Creamy egg and leek special",
    "subtitle" : "",
    "servings" : "4",
    "active" : "1",
    "parid" : "0",
    "userid" : "0",
    "rating" : "0.0",
    "cumrating" : "0.0",
    "createdate" : "0"
}
```

When preparing the hosts you must be aware of this translation of the different structures, as it will have an effect on the way the information is replicated from MySQL to MongoDB.

MySQL Host

The data replicated from MySQL can be any data, although there are some known limitations and assumptions made on the way the information is transferred.

When configuring the extractor database and host, ensure heterogenous specific prerequisites have been included, see Section B.4.4, “MySQL Configuration for Heterogeneous Deployments”

For the best results when replicating, be aware of the following issues and limitations:

- Use primary keys on all tables. The use of primary keys will improve the lookup of information within MongoDB when rows are updated. Without a primary key on a table a full table scan is performed, which can affect performance.
- MySQL `TEXT` columns are correctly replicated, but cannot be used as keys.
- MySQL `BLOB` columns are converted to text using the configured character type. Depending on the data that is being stored within the `BLOB`, the data may need to be custom converted. A filter can be written to convert and reformat the content as required.

MongoDB Host

- Enable networking; by default MongoDB is configured to listen only on the `localhost` (127.0.0.1) IP address. The address should be changed to the IP address off your host, or `0.0.0.0`, which indicates all interfaces on the current host.
- Ensure that network port 27017, or the port you want to use for MongoDB is configured as the listening port.

## 4.5.3. Install MongoDB Applier

### Note

The steps in this section relate specifically to applying to a standard MongoDB Instance. For configuring the applier to work with MongoDB Atlas, please refer to the following section: Section 4.5.4, “Install MongoDB Atlas Applier”

Installation of the MongoDB replication requires special configuration of the Source and Target hosts so that each is configured for the correct datasource type.

To configure the Applier replicators:

1. Before installing the applier, the following additions need adding to the extractor configuration. Apply the following parameter to the extractor configuration before installing the applier

   Add the following the `/etc/tungsten/tungsten.ini`

   ```
   [alpha]
   ...Existing Replicator Config...
   enable-heterogeneous-service=true

   shell> tpm update
   ```

### Note

> The above step is only applicable for standalone extractors. If you are configuring replications from an existing Tungsten Cluster (Cluster-Extractor), follow the steps outlined here to ensure the cluster is configured correctly: Section 3.4.1, "Prepare: Replicating Data Out of a Cluster"

2. Unpack the Tungsten Replicator distribution in staging directory:

```
shell> tar zxf tungsten-replicator-7.1.2-81.tar.gz
```

3. Change into the staging directory:

```
shell> cd tungsten-replicator-7.1.2-81
```

4. Configure the installation using tpm:

   Show Staging

   Show INI

```
shell> ./tools/tpm configure defaults \
    --reset \
    --install-directory=/opt/continuent \
    --profile-script=~/.bash_profile \
    --rest-api-admin-user=apiuser \
    --rest-api-admin-pass=secret

shell> ./tools/tpm configure alpha \
    --master=sourcehost \
    --members=localhost \
    --datasource-type=mongodb \
    --replication-user=tungsten \
    --replication-password=secret \
    --svc-applier-filters=dropstatementdata \
    --role=slave \
    --replication-port=27017
```

```
shell> vi /etc/tungsten/tungsten.ini
```

```
[defaults]
install-directory=/opt/continuent
profile-script=~/.bash_profile
rest-api-admin-user=apiuser
rest-api-admin-pass=secret

[alpha]
master=sourcehost
members=localhost
datasource-type=mongodb
replication-user=tungsten
replication-password=secret
svc-applier-filters=dropstatementdata
role=slave
replication-port=27017
```

   Configuration group `defaults`

   The description of each of the options is shown below; click the icon to hide this detail:

   Click the icon to show a detailed description of each argument.

   - `--reset` [422]

     `reset` [422]

     For staging configurations, deletes all pre-existing configuration information between updating with the new configuration values.

   - `--install-directory=/opt/continuent` [409]

     `install-directory=/opt/continuent` [409]

     Path to the directory where the active deployment will be installed. The configured directory will contain the software, THL and relay log information unless configured otherwise.

   - `--profile-script=~/.bash_profile` [417]

     `profile-script=~/.bash_profile` [417]

Append commands to include env.sh in this profile script

- `--rest-api-admin-user=apiuser` [422]

  `rest-api-admin-user=apiuser` [422]
  Optional: Must be specified along with rest-api-admin-pass if you wish to access the full API features and use the Dashboard GUI for cluster installations.

- `--rest-api-admin-pass=secret` [422]

  `rest-api-admin-pass=secret` [422]
  Optional: Must be specified along with rest-api-admin-user if you wish to access the full API features.

Configuration group **`alpha`**

The description of each of the options is shown below; click the icon to hide this detail:

Click the icon to show a detailed description of each argument.

- `--master=sourcehost` [412]

  `master=sourcehost` [412]

  The hostname of the primary (extractor) within the current service.

- `--members=localhost` [413]

  `members=localhost` [413]

  Hostnames for the dataservice members

- `--datasource-type=mongodb` [402]

  `datasource-type=mongodb` [402]

  Database type

- `--replication-user=tungsten` [421]

  `replication-user=tungsten` [421]

  For databases that required authentication, the username to use when connecting to the database using the corresponding connection method (native, JDBC, etc.).

- `--replication-password=secret` [420]

  `replication-password=secret` [420]

  The password to be used when connecting to the database using the corresponding `--replication-user` [421].

- `--svc-applier-filters=dropstatementdata` [424]

  `svc-applier-filters=dropstatementdata` [424]

  Replication service applier filters

- `--role=slave` [422]

  `role=slave` [422]

  What is the replication role for this service?

- `--replication-port=27017` [421]

  `replication-port=27017` [421]

  The network port used to connect to the database server. The default port used depends on the database being configured.

5. **Note**

   If you plan to make full use of the REST API (which is enabled by default) you will need to also configure a username and password for API access. This must be done by specifying the following options in your configuration:

   ```
   rest-api-admin-user=tungsten
   rest-api-admin-pass=secret
   ```

6. Once the prerequisites and configuring of the installation has been completed, the software can be installed:

   ```
   shell> ./tools/tpm install
   ```

If the installation process fails, check the output of the `/tmp/tungsten-configure.log` file for more information about the root cause.

Once the replicators have started, the status of the service can be checked using trepctl. See Section 4.5.5, "Management and Monitoring of MongoDB Deployments" for more information.

## 4.5.4. Install MongoDB Atlas Applier

**Note**

The steps in this section relate specifically to applying to a MongoDB Atlas Instance. For configuring the applier to work with standatd MongoDB, please refer to the following section: Section 4.5.3, "Install MongoDB Applier"

Installation of the MongoDB replication requires special configuration of the Source and Target hosts so that each is configured for the correct datasource type.

To configure the Applier replicators:

1. Before installing the applier, the following addition needs adding to the extractor configuration. Apply the following parameters on the extractor host, update the extractor using the details below, and then install the applier

   - For Staging installs:

     ```
     shell> cd tungsten-replicator-7.1.2-81
     shell> ./tools/tpm configure alpha \
     --enable-heterogeneous-master=true
     shell> ./tools/tpm update
     ```

   - For INI installs: Add the following the `/etc/tungsten/tungsten.ini`

     ```
     [alpha]
     ...Existing Replicator Config...
     enable-heterogeneous-master=true

     shell> tpm update
     ```

2. Unpack the Tungsten Replicator distribution in staging directory:

   ```
   shell> tar zxf tungsten-replicator-7.1.2-81.tar.gz
   ```

3. Change into the staging directory:

   ```
   shell> cd tungsten-replicator-7.1.2-81
   ```

4. Configure the installation using tpm:

   Show Staging

   Show INI

   ```
   shell> ./tools/tpm configure defaults \
       --reset \
       --install-directory=/opt/continuent \
       --profile-script=~/.bash_profile \
       --disable-security-controls=false \
       --rmi-ssl=false \
       --thl-ssl=false \
       --rmi-authentication=false \
       --rest-api-admin-user=apiuser \
       --rest-api-admin-pass=secret

   shell> ./tools/tpm configure alpha \
       --master=sourcehost \
       --members=localhost \
       --datasource-type=mongodb \
       --replication-user=tungsten \
   ```

```
    --replication-password=secret \
    --svc-applier-filters=dropstatementdata \
    --role=slave \
    --replication-host=atlasendpoint.mongodb.net \
    --replication-port=27017 \
    --property=replicator.applier.dbms.connectString=mongodb+srv://${replicator.global.db.user}:${replicator.global.db.password}@${replicator.global.db.host}/?r
```

```
shell> vi /etc/tungsten/tungsten.ini
```

```
[defaults]
install-directory=/opt/continuent
profile-script=~/.bash_profile
disable-security-controls=false
rmi-ssl=false
thl-ssl=false
rmi-authentication=false
rest-api-admin-user=apiuser
rest-api-admin-pass=secret

[alpha]
master=sourcehost
members=localhost
datasource-type=mongodb
replication-user=tungsten
replication-password=secret
svc-applier-filters=dropstatementdata
role=slave
replication-host=atlasendpoint.mongodb.net
replication-port=27017
property=replicator.applier.dbms.connectString=mongodb+srv://${replicator.global.db.user}:${replicator.global.db.password}@${replicator.global.db.host}/?retryWr
```

Configuration group **`defaults`**

The description of each of the options is shown below; click the icon to hide this detail:

Click the icon to show a detailed description of each argument.

- `--reset` [422]

  `reset` [422]

  For staging configurations, deletes all pre-existing configuration information between updating with the new configuration values.

- `--install-directory=/opt/continuent` [409]

  `install-directory=/opt/continuent` [409]

  Path to the directory where the active deployment will be installed. The configured directory will contain the software, THL and relay log information unless configured otherwise.

- `--profile-script=~/.bash_profile` [417]

  `profile-script=~/.bash_profile` [417]

  Append commands to include env.sh in this profile script

- `--disable-security-controls=false` [404]

  `disable-security-controls=false` [404]

  Disables all forms of security, including SSL, TLS and authentication

- `--rmi-ssl=false` [407]

  `rmi-ssl=false` [407]

  Enable SSL encryption of RMI communication on this host

- `--thl-ssl=false` [408]

  `thl-ssl=false` [408]

  Enable SSL encryption of THL communication for this service

- `--rmi-authentication=false` [407]

  `rmi-authentication=false` [407]

Enable RMI authentication for the services running on this host

- `--rest-api-admin-user=apiuser` [422]

  `rest-api-admin-user=apiuser` [422]
  Optional: Must be specified along with rest-api-admin-pass if you wish to access the full API features and use the Dashboard GUI for cluster installations.

- `--rest-api-admin-pass=secret` [422]

  `rest-api-admin-pass=secret` [422]
  Optional: Must be specified along with rest-api-admin-user if you wish to access the full API features.

Configuration group **`alpha`**

The description of each of the options is shown below; click the icon to hide this detail:

Click the icon to show a detailed description of each argument.

- `--master=sourcehost` [412]

  `master=sourcehost` [412]

  The hostname of the primary (extractor) within the current service.

- `--members=localhost` [413]

  `members=localhost` [413]

  Hostnames for the dataservice members

- `--datasource-type=mongodb` [402]

  `datasource-type=mongodb` [402]

  Database type

- `--replication-user=tungsten` [421]

  `replication-user=tungsten` [421]

  For databases that required authentication, the username to use when connecting to the database using the corresponding connection method (native, JDBC, etc.).

- `--replication-password=secret` [420]

  `replication-password=secret` [420]

  The password to be used when connecting to the database using the corresponding `--replication-user` [421].

- `--svc-applier-filters=dropstatementdata` [424]

  `svc-applier-filters=dropstatementdata` [424]

  Replication service applier filters

- `--role=slave` [422]

  `role=slave` [422]

  What is the replication role for this service?

- `--replication-host=atlasendpoint.mongodb.net` [420]

  `replication-host=atlasendpoint.mongodb.net` [420]

  Hostname of the datasource where the database is located. If the specified hostname matches the current host or member name, the database is assumed to be local. If the hostnames do not match, extraction is assumed to be via remote access. For MySQL hosts, this configures a remote replication Replica (relay) connection.

- `--replication-port=27017` [421]

`replication-port=27017` [421]

The network port used to connect to the database server. The default port used depends on the database being configured.

- `--property=replicator.applier.dbms.connectString=mongodb+srv://${replicator.global.db.user}:${replicator.global.db.password}@${replicator.global.db.host}/?retryWrites=true&w=majority` [368]

  `property=replicator.applier.dbms.connectString=mongodb+srv://${replicator.global.db.user}:${replicator.global.db.password}@${replicator.global.db.host}/?retryWrites=true&w=majority` [368]

  The `--property` [368] option enables you to explicitly set property values in the target files. A number of different models are supported:

  - `key=value`

    Set the property defined by `key` to the specified value without evaluating any template values or other rules.

  - `key+=value`

    Add the value to the property defined by `key`. Template values and other options append their settings to the end of the specified property.

  - `key~=/match/replace/`

    Evaluate any template values and other settings, and then perform the specified Ruby regex operation to the property defined by `key`. For example `--property=replicator.key~=/(.*)/somevalue,\1/` will prepend `somevalue` before the template value for `replicator.key`.

5. **Note**

   > If you plan to make full use of the REST API (which is enabled by default) you will need to also configure a username and password for API access. This must be done by specifying the following options in your configuration:
   >
   > ```
   > rest-api-admin-user=tungsten
   > rest-api-admin-pass=secret
   > ```

6. Once the prerequisites and configuring of the installation has been completed, the software can be installed:

```
shell> ./tools/tpm install
```

If the installation process fails, check the output of the `/tmp/tungsten-configure.log` file for more information about the root cause.

**Important**

> The above example assumes SSL is not enabled between the extractor and applier replicators.
>
> If SSL is required, then you must omit the following properties from the example configs displayed above, or change the values to `true`: `rmi-ssl=false`, `thl-ssl=false`, `rmi-authentication=false`

Once you have installed the replicator, there are a few more steps required to allow the replicator to be able to authenticate with MongoDB Atlas.

### 4.5.4.1. Import MongoDB Atlas Certificates

MongoDB Atlas requires TLS connections for all Atlas Clusters, therefore we need to configure the replicator to recognise this.

**Note**

> From May 1, 2021, MongoDB Atlas has moved to new TLS Certificiates using ISRG instead of IdenTrust for their root Certificate Authority.
>
> All new clusters created after this time, or any existing clusters that have since been migrated to this new root CA will need to follow the correct procedure to configure the replicator. Both procedures are below, follow the correct one that relates to your configuration.

For MongoDB Atlas Cluster created PRIOR to May 1, 2021, or that have not yet migrated to the new LetsEncrypt root Certificate:

1. Using the correct Atlas Endpoint, issue the following command to retrieve the Atlas certificates

```
shell> openssl s_client -showcerts -connect atlas-endpoint.mongodb.net:27017
```

2. The output may be quite long and will include at least two certificates bound by the header/footer as follows

```
-----BEGIN CERTIFICATE-----
xxxx
xxxx
-----END CERTIFICATE-----
```

   Copy each certificate, including the header/footer, into individual files

3. Using keytool, we now need to load each certificte into the truststore that was created during the replicator installation. Repeat the example below for each certificate, ensuring you use a unique alias name for each certificate.

```
shell> keytool -import -alias your-alias1 -file cert1.cer -keystore /opt/continuent/share/tungsten_truststore.ts
```

   When prompted, the default password for the truststore will be `tungsten` unless you specified a different password during installation

4. Once this is complete, you can now start the replicator

```
shell> replicator start
```

For MongoDB Atlas Cluster created AFTER May 1, 2021, or that have been migrated to the new LetsEncrypt root Certificate:

1. Obtain the LetsEncrypt root Certificate from here

2. Copy the certificate into a file called `letsencrypt.pem` in the home directory of the applier host, including the BEGIN an END header/footer, for example:

```
-----BEGIN CERTIFICATE-----
xxxx
xxxx
-----END CERTIFICATE-----
```

3. Using keytool, we now need to import this certificte into the truststore that was created during the replicator installation.

```
shell> keytool -import -alias letsencrypt -file letsencrypt.pem -keystore /opt/continuent/share/tungsten_truststore.ts
```

   When prompted, the default password for the truststore will be `tungsten` unless you specified a different password during installation

4. Once this is complete, you can now start the replicator

```
shell> replicator start
```

Once the replicators have started, the status of the service can be checked using trepctl. See Section 4.5.5, “Management and Monitoring of MongoDB Deployments” for more information.

## 4.5.5. Management and Monitoring of MongoDB Deployments

Once the two services — extractor and applier — have been installed, the services can be monitored using trepctl. To monitor the extractor service:

```
shell> trepctl status
Processing status command...
NAME                     VALUE
----                     -----
appliedLastEventId     : mysql-bin.000008:0000000000412301;0
appliedLastSeqno       : 1296
appliedLatency         : 1.889
channels               : 1
clusterName            : epsilon
currentEventId         : mysql-bin.000008:0000000000412301
currentTimeMillis      : 1377097812795
dataServerHost         : host1
extensions             :
latestEpochNumber      : 1286
masterConnectUri       : thl://localhost:/
masterListenUri        : thl://host2:2112/
maximumStoredSeqNo     : 1296
minimumStoredSeqNo     : 0
offlineRequests        : NONE
pendingError           : NONE
pendingErrorCode       : NONE
pendingErrorEventId    : NONE
pendingErrorSeqno      : -1
pendingExceptionMessage: NONE
pipelineSource         : jdbc:mysql:thin://host1:13306/
relativeLatency        : 177444.795
```

```
resourcePrecedence     : 99
rmiPort                : 10000
role                   : master
seqnoType              : java.lang.Long
serviceName            : alpha
serviceType            : local
simpleServiceName      : alpha
siteName               : default
sourceId               : host1
state                  : ONLINE
timeInStateSeconds     : 177443.948
transitioningTo        :
uptimeSeconds          : 177461.483
version                : Tungsten Replicator 7.1.2 build 81
Finished status command...
```

The replicator service operates just the same as a standard Extractor service of a typical MySQL replication service.

The MongoDB applier service can be accessed either remotely from the Extractor:

```
shell> trepctl -host host2 status
...
```

Or locally on the MongoDB host:

```
shell> trepctl status
Processing status command...
NAME                     VALUE
----                     -----
appliedLastEventId     : mysql-bin.000008:0000000000412301;0
appliedLastSeqno       : 1296
appliedLatency         : 10.253
channels               : 1
clusterName            : alpha
currentEventId         : NONE
currentTimeMillis      : 1377098139212
dataServerHost         : host2
extensions             :
latestEpochNumber      : 1286
masterConnectUri       : thl://host1:2112/
masterListenUri        : null
maximumStoredSeqNo     : 1296
minimumStoredSeqNo     : 0
offlineRequests        : NONE
pendingError           : NONE
pendingErrorCode       : NONE
pendingErrorEventId    : NONE
pendingErrorSeqno      : -1
pendingExceptionMessage: NONE
pipelineSource         : thl://host1:2112/
relativeLatency        : 177771.212
resourcePrecedence     : 99
rmiPort                : 10000
role                   : slave
seqnoType              : java.lang.Long
serviceName            : alpha
serviceType            : local
simpleServiceName      : alpha
siteName               : default
sourceId               : host2
state                  : ONLINE
timeInStateSeconds     : 177783.343
transitioningTo        :
uptimeSeconds          : 180631.276
version                : Tungsten Replicator 7.1.2 build 81
Finished status command...
```

Monitoring the status of replication between the Source and Target is also the same. The `appliedLastSeqno` still indicates the sequence number that has been applied to MongoDB, and the event ID from MongoDB can still be identified from `appliedLastEventId`.

Sequence numbers between the two hosts should match, as in a Primary/Replica deployment, but due to the method used to replicate, the applied latency may be higher. Tables that do not use primary keys, or large individual row updates may cause increased latency differences.

To check for information within MongoDB, use the mongo command-line client:

```
shell> mongo
MongoDB shell version: 2.2.4
connecting to: test
> use cheffy;
switched to db cheffy
```

The show collections will indicate the tables from MySQL that have been replicated to MongoDB:

```
> show collections
access_log
audit_trail
blog_post_record
helpdb
ingredient_recipes
ingredient_recipes_bytext
ingredients
ingredients_alt
ingredients_keywords
ingredients_matches
ingredients_measures
ingredients_plurals
ingredients_search_class
ingredients_search_class_map
ingredients_shop_class
ingredients_xlate
ingredients_xlate_class
keyword_class
keywords
measure_plurals
measure_trans
metadata
nut_fooddesc
nut_foodgrp
nut_footnote
nut_measure
nut_nutdata
nut_nutrdef
nut_rda
nut_rda_class
nut_source
nut_translate
nut_weight
recipe
recipe_coll_ids
recipe_coll_search
recipe_collections
recipe_comments
recipe_pics
recipebase
recipeingred
recipekeywords
recipemeta
recipemethod
recipenutrition
search_translate
system.indexes
terms
```

Collection counts should match the row count of the source tables:

```
> > db.recipe.count()
2909
```

The db.collection.find() command can be used to list the documents within a given collection.

```
> db.recipe.find()
{ "_id" : ObjectId("5212233584ae46ce07e427c3"),
"recipeid" : "1085",
"title" : "Creamy egg and leek special",
"subtitle" : "",
"servings" : "4",
"active" : "1",
"parid" : "0",
"userid" : "0",
"rating" : "0.0",
"cumrating" : "0.0",
"createdate" : "0" }
{ "_id" : ObjectId("5212233584ae46ce07e427c4"),
 "recipeid" : "87",
 "title" : "Chakchouka",
 "subtitle" : "A traditional Arabian and North African dish and often accompanied with slices of cooked meat",
 "servings" : "4",
 "active" : "1",
 "parid" : "0",
 "userid" : "0",
 "rating" : "0.0",
 "cumrating" : "0.0",
 "createdate" : "0" }
 ...
```

The output should be checked to ensure that information is being correctly replicated. If strings are shown as a hex value, for example:

```
"title" : "[B@7084a5c"
```

It probably indicates that UTF8 and/or `--mysql-use-bytes-for-string=false` [414] options were not used during installation. The configuration can be updated using tpm to address this issue.

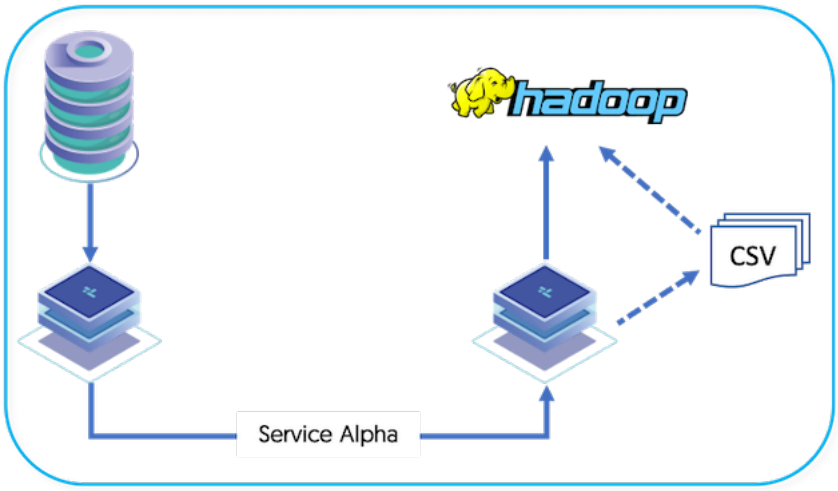
# 4.6. Deploying the Hadoop Applier

Replicating data into Hadoop is achieved by generating character-separated values from ROW-based information that is applied directly to the Hadoop HDFS using a batch loading process. Files are written directly to the HDFS using the Hadoop client libraries. A separate process is then used to merge existing data, and the changed information extracted from the Source database.

Deployment of the Hadoop replication is similar to other heterogeneous installations; two separate installations are created:

- Service Alpha on the extractor, extracts the information from the MySQL binary log into THL.
- Service Alpha on the applier, reads the information from the remote replicator as THL, applying it to Hadoop. The applier works in two stages:

Figure 4.7. Topologies: Replicating to Hadoop



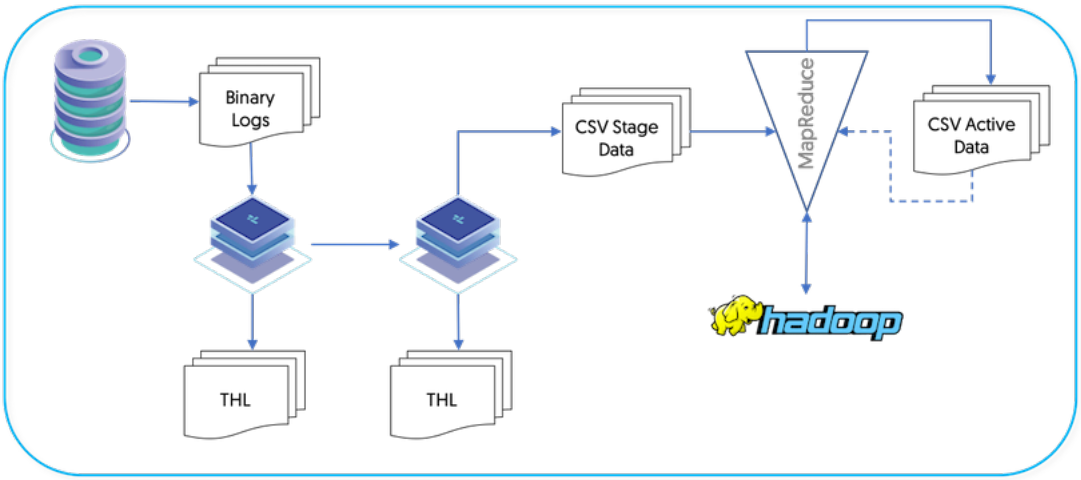

Basic requirements for replication into Hadoop:

- Hadoop Replication is supported on the following Hadoop distributions and releases:
  - Cloudera Enterprise 4.4, Cloudera Enterprise 5.0 (Certified) up to Cloudera Enterprise 5.8
  - HortonWorks DataPlatform 2.0
  - Amazon Elastic MapReduce
  - IBM InfoSphere BigInsights 2.1 and 3.0
  - MapR 3.0, 3.1, and 5.x
  - Pivotal HD 2.0
  - Apache Hadoop 2.1.0, 2.2.0
- Source tables must have primary keys. Without a primary key, Tungsten Replicator is unable to determine the row to be updated when the data reaches Hadoop.

## 4.6.1. Hadoop Replication Operation

The Hadoop applier makes use of the JavaScript based batch loading system (see Section 5.6.4, “JavaScript Batchloader Scripts”). This constructs change data from the source-database, and uses this information in combination with any existing data to construct, using Hive, a materialized view. A summary of this basic structure can be seen in Figure 4.8, “Topologies: Hadoop Replication Operation”.

Figure 4.8. Topologies: Hadoop Replication Operation




The full replication of information operates as follows:

1. Data is extracted from the source database using the standard extractor, for example by reading the row change data from the binlog in MySQL.

2. The `colnames` filter is used to extract column name information from the database. This enables the row-change information to be tagged with the corresponding column information. The data changes, and corresponding row names, are stored in the THL.

   The `pkey` filter is used to extract primary key data from the source tables.

3. On the applier replicator, the THL data is read and written into batch-files in the character-separated value format.

   The information in these files is change data, and contains not only the original data, but also metadata about the operation performed (i.e. `INSERT`, `DELETE` or `UPDATE`, and the primary key of for each table. All `UPDATE` statements are recorded as a `DELETE` of the existing data, and an `INSERT` of the new data.

4. A second process uses the CSV stage data and any existing data, to build a materialized view that mirrors the source table data structure.

The staging files created by the replicator are in a specific format that incorporates change and operation information in addition to the original row data.

- The format of the files is a character separated values file, with each row separated by a newline, and individual fields separated by the character `0x01`. This is supported by Hive as a native value separator.

- The content of the file consists of the full row data extracted from the source, plus metadata describing the operation for each row, the sequence number, and then the full row information.

| Operation | Sequence No | Unique Row | Commit TimeStamp | Table-specific primary key | Table-column |
|---|---|---|---|---|---|
| I (Insert) or D (Delete) | *SEQNO* [549] that generated this row | Unique row ID within the batch | The commit timestamp of the original transaction, which can be used for partitioning | | |

For example, the MySQL row:

```
|  3 | #1 Single | 2006 | Cats and Dogs (#1.4)         |
```

Is represented within the staging files generated as:

```
I^A1318^A1^A2017-06-07 09:22:28.000^A3^A3^A#1 Single^A2006^ACats and Dogs (#1.4)
```

The character separator, and whether to use quoting, are configurable within the replicator when it is deployed. The default is to use a newline character for records, and the `0x01` character for fields. For more information on these fields and how they can be configured, see Section 5.6.7, “Supported CSV Formats”.

On the Hadoop host, information is stored into a number of locations within the HDFS during the data transfer:

Table 4.2. Hadoop Replication Directory Locations

| Directory/File | Description |
|---|---|
| `/user/USERNAME` | Top-level directory for Tungsten Replicator information, using the configured replication user. |
| `/user/tungsten/metadata` | Location for metadata related to the replication operation |
| `/user/tungsten/metadata/alpha` | The directory (named after the servicename of the replicator service) that holds service-specific metadata |
| `/user/tungsten/staging` | Directory of the data transferred |
| `/user/tungsten/staging/servicename` | Directory of the data transferred from a specific servicename. |
| `/user/tungsten/staging/servicename/databasename` | Directory of the data transferred specific to a database. |
| `/user/tungsten/staging/servicename/databasename/tablename` | Directory of the data transferred specific to a table. |
| `/user/tungsten/staging/servicename/databasename/tablename/tablename-###.csv` | Filename of a single file of the data transferred for a specific table and database. |

Files are automatically created, named according to the parent table name, and the starting Tungsten Replicator sequence number for each file that is transferred. The size of the files is determined by the batch and commit parameters. For example, in the truncated list of files below displayed using the hadoop fs command,

```
shell> hadoop fs -ls /user/tungsten/staging/hadoop/chicago
Found 66 items
-rw-r--r-- 3 cloudera cloudera  1270236 2020-01-13 06:58 /user/tungsten/staging/alpha/hadoop/chicago/chicago-10.csv
-rw-r--r-- 3 cloudera cloudera 10274189 2020-01-13 08:33 /user/tungsten/staging/alpha/hadoop/chicago/chicago-103.csv
-rw-r--r-- 3 cloudera cloudera  1275832 2020-01-13 08:33 /user/tungsten/staging/alpha/hadoop/chicago/chicago-104.csv
-rw-r--r-- 3 cloudera cloudera  1275411 2020-01-13 08:33 /user/tungsten/staging/alpha/hadoop/chicago/chicago-105.csv
-rw-r--r-- 3 cloudera cloudera 10370471 2020-01-13 08:33 /user/tungsten/staging/alpha/hadoop/chicago/chicago-113.csv
-rw-r--r-- 3 cloudera cloudera  1279435 2020-01-13 08:33 /user/tungsten/staging/alpha/hadoop/chicago/chicago-114.csv
-rw-r--r-- 3 cloudera cloudera  2544062 2020-01-13 06:58 /user/tungsten/staging/alpha/hadoop/chicago/chicago-12.csv
-rw-r--r-- 3 cloudera cloudera 11694202 2020-01-13 08:33 /user/tungsten/staging/alpha/hadoop/chicago/chicago-123.csv
-rw-r--r-- 3 cloudera cloudera  1279072 2020-01-13 08:34 /user/tungsten/staging/alpha/hadoop/chicago/chicago-124.csv
-rw-r--r-- 3 cloudera cloudera  2570481 2020-01-13 08:34 /user/tungsten/staging/alpha/hadoop/chicago/chicago-126.csv
-rw-r--r-- 3 cloudera cloudera  9073627 2020-01-13 08:34 /user/tungsten/staging/alpha/hadoop/chicago/chicago-133.csv
-rw-r--r-- 3 cloudera cloudera  1279708 2020-01-13 08:34 /user/tungsten/staging/alpha/hadoop/chicago/chicago-134.csv
...
```

The individual file numbers will not be sequential, as they will depend on the sequence number, batch size and range of tables transferred.

## 4.6.2. Preparing for Hadoop Replication

During the replication process, data is exchanged from the MySQL database/table/row structure into corresponding Hadoop directory and files, as shown in the table below:

| MySQL | Hadoop |
|---|---|
| Database | Directory |
| Table | Hive-compatible Character-Separated Text file |
| Row | Line in the text file, fields terminated by character `0x01` |

### 4.6.2.1. Hadoop Host

The Hadoop environment should have the following features and parameters for the most efficient operation:

- Disk storage

  There must be enough disk storage for the change data, data being actively merged, and the live data for the replicated information. Depending on the configuration and rate of changes in the Source, the required data space will fluctuate.

  For example, replicating a 10GB dataset, and 5GB of change data during replication, will require at least 30GB of storage. 10GB for the original dataset, 5GB of change data, and 10-25GB of merged data. The exact size is dependent on the quantity of inserts/updates/deletes.

- Pre-requisites

Currently, deployment of the target to a relay host is not supported. One host within the Hadoop cluster must be chosen to act as the target.

The prerequisites for a standard Tungsten Replicator should be followed, including:

- Section B.3.1, "Creating the User Environment"
- Section B.3.2.1, "Network Ports"
- Section B.3.3, "Directory Locations and Configuration"
- Section B.3.4, "Configure Software"

This will provide the base environment into which Tungsten Replicator can be installed.

- HDFS Location

  The `/user/tungsten` directory must be writable by the replicator user within HDFS:

  ```
  shell> hadoop fs -mkdir /user/tungsten
  shell> hadoop fs -chmod 700 /user/tungsten
  shell> hadoop fs -chown tungsten /user/tungsten
  ```

  These commands should be executed by a user with HDFS administration rights (e.g. the `hdfs` user).

- Replicator User Group Membership

  The user that will be executing the replicator (typically `tungsten`, as recommended in the Appendix B, *Prerequisites*) must be a member of the `hive` group on the Hadoop host where the replicator will be installed. Without this membership, the user will be unable to execute Hive queries.

### 4.6.2.2. Schema Generation

In order to access the generated tables, both staging and the final tables, it is necessary to create a schema definition. The ddlscan tool can be used to read the existing definition of the tables from the source server and generate suitable Hive schema definitions to access the table data.

To create the staging table definition, use the `ddl-mysql-hive-0.10.vm` template; you must specify the JDBC connection string, user, password and database names. For example:

```
shell> ddlscan -user tungsten -url 'jdbc:mysql:thin://host1:13306/test' -pass password \
   -template ddl-mysql-hive-0.10.vm -db test
--
-- SQL generated on Wed Jan 29 16:17:05 GMT 2020 by Tungsten ddlscan utility
--
-- url = jdbc:mysql:thin://host1:13306/test
-- user = tungsten
-- dbName = test
--
CREATE DATABASE test;

DROP TABLE IF EXISTS test.movies_large;

CREATE TABLE test.movies_large
(
  id INT ,
  title STRING ,
  year INT ,
  episodetitle STRING  )
;
```

The output from this command should be applied to your Hive installation within the Hadoop cluster. For example, by capturing the output, transferring that file and then running:

```
shell> cat schema.sql | hive
```

To create Hive tables that read the staging files loaded by the replicator, use the `ddl-mysql-hive-0.10-staging.vm`:

```
shell> ddlscan -user tungsten -url 'jdbc:mysql:thin://host:13306/test' -pass password \
    -template ddl-mysql-hive-0.10-staging.vm -db test
```

The process creates the schema and tables which match the schema and table names on the source database.

Transfer this file to your Hadoop environment and then create the generated schema:

```
shell> cat schema-staging.sql |hive
```

The process creates matching schema names, but table names are modified to include the prefix `stage_xxx_`. For example, for the table `movies_large` a staging table named `stage_xxx_movies_large` is created. The Hive table definition is created pointing to the external file-based tables, using the default `0x01` field separator and `0x0A` (newline) record separator. If different values were used for these in the configuration, the schema definition in the captured file from ddlscan should be updated by hand.

The tables should now be available within Hive. For more information on accessing and using the tables, see Section 4.6.4.3, "Accessing Generated Tables in Hive".

## 4.6.3. Replicating into Kerberos Secured HDFS

For replicating into HDFS where Kerberos support has been enabled, the `hadoop_kerberos.js` vatch script can be used in place of the normal `hadoop.js` script.

The script will need modification before it can be used, due to the varying implementations of Kerberos, and to ensure the correct authentication parameters are used.

Before installed, edit the `hadoop_kerberos.js` file located within `tungsten-replicator/appliers/batch/hadoop-kerberos.js` within the installation package. Within that file is the line called before the HDFS operations are called:

```
var kinit_prefix = "kinit USER/LEVEL@REALM -k -t KEYTAB_FILE;"
```

Edit this line to set the correct command and/or authentication parameters, such as the username and keytab file. The configured command will be executed immediately before all the commands that operate on the Hadoop filesystem, including creating directories and files.

For example, the variable might be updated to:

```
var kinit_prefix = "kinit mc/admin@CLOUDERA -k -t mcadmin.keytab;"
```

When installing, use `--batch-load-template=hadoop_kerberos.js` [398] to enable the new batch load script.

## 4.6.4. Install Hadoop Replication

Installation of the Hadoop replication consists of multiple stages:

1. Configure the source and target hosts following the prerequisites outlined in Appendix B, *Prerequisites* then follow the appropriate steps for the required extractor topology outlined in Chapter 3, *Deploying MySQL Extractors*.
2. Install the Applier replicator which will apply information to the target Hadoop environment.
3. Once the installation of the Extractor and Applier components have been completed, materialization of tables and views can be performed.

### 4.6.4.1. Applier Replicator Service

The applier replicator service reads information from the THL of the source and applies this to a local instance of Hadoop.

> **Important**
>
> Installation must take place on a node within the Hadoop cluster. Writing to a remote HDFS filesystem is not currently supported.

1. Before installing the applier, the following additions need adding to the extractor configuration. Apply the following parameters, update the extractor and then install the applier
   - For Staging Install:

     ```
     shell> cd tungsten-replicator-7.1.2-81
     shell> ./tools/tpm configure alpha \
       --enable-batch-service=true
     shell> ./tools/tpm update
     ```

   - For INI Installs: Add the following the `/etc/tungsten/tungsten.ini`

     ```
     [alpha]
     ...Existing Replicator Config...
     enable-batch-service=true


     shell> tpm update
     ```

2. The applier can now be configured.

   Unpack the Tungsten Replicator distribution in staging directory:

```
shell> tar zxf tungsten-replicator-7.1.2-81.tar.gz
```

3. Change into the staging directory:

```
shell> cd tungsten-replicator-7.1.2-81
```

4. Configure the installation using tpm:

   Show Staging

   Show INI

```
shell> ./tools/tpm configure defaults \
    --reset \
    --user=tungsten \
    --install-directory=/opt/continuent \
    --profile-script=~/.bash_profile \
    --skip-validation-check=HostsFileCheck \
    --skip-validation-check=InstallerMasterSlaveCheck \
    --skip-validation-check=DatasourceDBPort \
    --skip-validation-check=DirectDatasourceDBPort \
    --skip-validation-check=ReplicationServicePipelines \
    --rest-api-admin-user=apiuser \
    --rest-api-admin-pass=secret

shell> ./tools/tpm configure alpha \
    --master=host1 \
    --members=host2 \
    --property=replicator.datasource.global.csvType=hive \
    --property=replicator.stage.q-to-dbms.blockCommitInterval=1s \
    --property=replicator.stage.q-to-dbms.blockCommitRowCount=1000 \
    --replication-password=secret \
    --replication-user=tungsten \
    --batch-enabled=true \
    --batch-load-language=js  \
    --batch-load-template=hadoop \
    --datasource-type=file
```

```
shell> vi /etc/tungsten/tungsten.ini
```

```
[defaults]
user=tungsten
install-directory=/opt/continuent
profile-script=~/.bash_profile
skip-validation-check=HostsFileCheck
skip-validation-check=InstallerMasterSlaveCheck
skip-validation-check=DatasourceDBPort
skip-validation-check=DirectDatasourceDBPort
skip-validation-check=ReplicationServicePipelines
rest-api-admin-user=apiuser
rest-api-admin-pass=secret

[alpha]
master=host1
members=host2
property=replicator.datasource.global.csvType=hive
property=replicator.stage.q-to-dbms.blockCommitInterval=1s
property=replicator.stage.q-to-dbms.blockCommitRowCount=1000
replication-password=secret
replication-user=tungsten
batch-enabled=true
batch-load-language=js
batch-load-template=hadoop
datasource-type=file
```

   Configuration group `defaults`

   The description of each of the options is shown below; click the icon to hide this detail:

   Click the icon to show a detailed description of each argument.

   - `--reset` [422]

     `reset` [422]

     For staging configurations, deletes all pre-existing configuration information between updating with the new configuration values.

- `--user=tungsten` [429]

  `user=tungsten` [429]

  System User

- `--install-directory=/opt/continuent` [409]

  `install-directory=/opt/continuent` [409]

  Path to the directory where the active deployment will be installed. The configured directory will contain the software, THL and relay log information unless configured otherwise.

- `--profile-script=~/.bash_profile` [417]

  `profile-script=~/.bash_profile` [417]

  Append commands to include env.sh in this profile script

- `--skip-validation-check=HostsFileCheck` [369]

  `skip-validation-check=HostsFileCheck` [369]

  The `--skip-validation-check` [369] disables a given validation check. If any validation check fails, the installation, validation or configuration will automatically stop.

  > **Warning**
  >
  > Using this option enables you to bypass the specified check, although skipping a check may lead to an invalid or non-working configuration.

  You can identify a given check if an error or warning has been raised during configuration. For example, the default table type check:

  ```
  ...
  ERROR >> centos >> The datasource root@centos:3306 (WITH PASSWORD) »
   uses MyISAM as the default storage engine (MySQLDefaultTableTypeCheck)
  ...
  ```

  The check in this case is `MySQLDefaultTableTypeCheck` [380], and could be ignored using `--skip-validation-check=MySQLDefaultTableTypeCheck` [369].

  Setting both `--skip-validation-check` [369] and `--enable-validation-check` [367] is equivalent to explicitly disabling the specified check.

- `--skip-validation-check=InstallerMasterSlaveCheck` [369]

  `skip-validation-check=InstallerMasterSlaveCheck` [369]

  The `--skip-validation-check` [369] disables a given validation check. If any validation check fails, the installation, validation or configuration will automatically stop.

  > **Warning**
  >
  > Using this option enables you to bypass the specified check, although skipping a check may lead to an invalid or non-working configuration.

  You can identify a given check if an error or warning has been raised during configuration. For example, the default table type check:

  ```
  ...
  ERROR >> centos >> The datasource root@centos:3306 (WITH PASSWORD) »
   uses MyISAM as the default storage engine (MySQLDefaultTableTypeCheck)
  ...
  ```

  The check in this case is `MySQLDefaultTableTypeCheck` [380], and could be ignored using `--skip-validation-check=MySQLDefaultTableTypeCheck` [369].

  Setting both `--skip-validation-check` [369] and `--enable-validation-check` [367] is equivalent to explicitly disabling the specified check.

- `--skip-validation-check=DatasourceDBPort` [369]

  `skip-validation-check=DatasourceDBPort` [369]

The `--skip-validation-check` [369] disables a given validation check. If any validation check fails, the installation, validation or configuration will automatically stop.

> **Warning**
>
> Using this option enables you to bypass the specified check, although skipping a check may lead to an invalid or non-working configuration.

You can identify a given check if an error or warning has been raised during configuration. For example, the default table type check:

```
...
ERROR >> centos >> The datasource root@centos:3306 (WITH PASSWORD) »
 uses MyISAM as the default storage engine (MySQLDefaultTableTypeCheck)
...
```

The check in this case is `MySQLDefaultTableTypeCheck` [380], and could be ignored using `--skip-validation-check=MySQLDefaultTableTypeCheck` [369].

Setting both `--skip-validation-check` [369] and `--enable-validation-check` [367] is equivalent to explicitly disabling the specified check.

- `--skip-validation-check=DirectDatasourceDBPort` [369]

  `skip-validation-check=DirectDatasourceDBPort` [369]

  The `--skip-validation-check` [369] disables a given validation check. If any validation check fails, the installation, validation or configuration will automatically stop.

  > **Warning**
  >
  > Using this option enables you to bypass the specified check, although skipping a check may lead to an invalid or non-working configuration.

  You can identify a given check if an error or warning has been raised during configuration. For example, the default table type check:

  ```
  ...
  ERROR >> centos >> The datasource root@centos:3306 (WITH PASSWORD) »
   uses MyISAM as the default storage engine (MySQLDefaultTableTypeCheck)
  ...
  ```

  The check in this case is `MySQLDefaultTableTypeCheck` [380], and could be ignored using `--skip-validation-check=MySQLDefaultTableTypeCheck` [369].

  Setting both `--skip-validation-check` [369] and `--enable-validation-check` [367] is equivalent to explicitly disabling the specified check.

- `--skip-validation-check=ReplicationServicePipelines` [369]

  `skip-validation-check=ReplicationServicePipelines` [369]

  The `--skip-validation-check` [369] disables a given validation check. If any validation check fails, the installation, validation or configuration will automatically stop.

  > **Warning**
  >
  > Using this option enables you to bypass the specified check, although skipping a check may lead to an invalid or non-working configuration.

  You can identify a given check if an error or warning has been raised during configuration. For example, the default table type check:

  ```
  ...
  ERROR >> centos >> The datasource root@centos:3306 (WITH PASSWORD) »
   uses MyISAM as the default storage engine (MySQLDefaultTableTypeCheck)
  ...
  ```

  The check in this case is `MySQLDefaultTableTypeCheck` [380], and could be ignored using `--skip-validation-check=MySQLDefaultTableTypeCheck` [369].

  Setting both `--skip-validation-check` [369] and `--enable-validation-check` [367] is equivalent to explicitly disabling the specified check.

- `--rest-api-admin-user=apiuser` [422]

  `rest-api-admin-user=apiuser` [422]

Optional: Must be specified along with rest-api-admin-pass if you wish to access the full API features and use the Dashboard GUI for cluster installations.

- `--rest-api-admin-pass=secret` [422]

  `rest-api-admin-pass=secret` [422]
  Optional: Must be specified along with rest-api-admin-user if you wish to access the full API features.

Configuration group **`alpha`**

The description of each of the options is shown below; click the icon to hide this detail:

Click the icon to show a detailed description of each argument.

- `--master=host1` [412]

  `master=host1` [412]

  The hostname of the primary (extractor) within the current service.

- `--members=host2` [413]

  `members=host2` [413]

  Hostnames for the dataservice members

- `--replication-password=secret` [420]

  `replication-password=secret` [420]

  The password to be used when connecting to the database using the corresponding `--replication-user` [421].

- `--replication-user=tungsten` [421]

  `replication-user=tungsten` [421]

  For databases that required authentication, the username to use when connecting to the database using the corresponding connection method (native, JDBC, etc.).

- `--batch-enabled=true` [397]

  `batch-enabled=true` [397]

  Should the replicator service use a batch applier

- `--batch-load-language=js` [398]

  `batch-load-language=js` [398]

  Which script language to use for batch loading

- `--batch-load-template=hadoop` [398]

  `batch-load-template=hadoop` [398]

  Value for the loadBatchTemplate property

- `--datasource-type=file` [402]

  `datasource-type=file` [402]

  Database type

5. **Note**

   If you plan to make full use of the REST API (which is enabled by default) you will need to also configure a username and password for API access. This must be done by specifying the following options in your configuration:

   ```
   rest-api-admin-user=tungsten
   rest-api-admin-pass=secret
   ```

6. Once the prerequisites and configuring of the installation has been completed, the software can be installed:

```
shell> ./tools/tpm install
```

If the installation process fails, check the output of the `/tmp/tungsten-configure.log` file for more information about the root cause.

Once the service has been installed it can be monitored using the trepctl command. See Section 4.6.4.4, "Management and Monitoring of Hadoop Deployments" for more information. If there are problems during installation, see Section 4.6.4.5, "Troubleshooting Hadoop Replication".

### 4.6.4.2. Generating Materialized Views

Added in 6.0.4. From Tungsten Replicator 6.0.4, continuent-tools-hadoop are now packaged within the main Tungsten Replicator software bundle and can be found within ./tungsten-replicator/support/hadoop-tools

The `continuent-tools-hadoop` repository contains a set of tools that allow for the convenient creation of DDL, materialized views, and data comparison on the tables that have been replicated from MySQL.

To obtain the tools, use git

```
shell> ./bin/load-reduce-check -s test -Ujdbc:mysql:thin://tr-hadoop2:13306 -udbload -ppassword
```

The load-reduce-check command performs four distinct steps:

1. Reads the schema from the MySQL server and creates the staging table DDL within Hive
2. Reads the schema from the MySQL server and creates the base table DDL within Hive
3. Executes the materialized view process on each selected staging table data to build the base table content.
4. Performs a data comparison

### 4.6.4.3. Accessing Generated Tables in Hive

If not already completed, the schema generation process described in Section 4.6.2.2, "Schema Generation" should have been followed. This creates the necessary Hive schema and staging schema definitions.

Once the tables have been created through ddlscan you can query the stage tables:

```
hive> select * from stage_xxx_movies_large limit 10;
OK
I 10 1 57475 All in the Family 1971 Archie Feels Left Out (#4.17)
I 10 2 57476 All in the Family 1971 Archie Finds a Friend (#6.18)
I 10 3 57477 All in the Family 1971 Archie Gets the Business: Part 1 (#8.1)
I 10 4 57478 All in the Family 1971 Archie Gets the Business: Part 2 (#8.2)
I 10 5 57479 All in the Family 1971 Archie Gives Blood (#1.4)
I 10 6 57480 All in the Family 1971 Archie Goes Too Far (#3.17)
I 10 7 57481 All in the Family 1971 Archie in the Cellar (#4.10)
I 10 8 57482 All in the Family 1971 Archie in the Hospital (#3.15)
I 10 9 57483 All in the Family 1971 Archie in the Lock-Up (#2.3)
I 10 10 57484 All in the Family 1971 Archie Is Branded (#3.20)
```

### 4.6.4.4. Management and Monitoring of Hadoop Deployments

Once the two services — extractor and applier — have been installed, the services can be monitored using trepctl. To monitor the Extractor service:

```
shell>  trepctl status
Processing status command...
NAME                     VALUE
----                     -----
appliedLastEventId     : mysql-bin.000023:0000000505545003;0
appliedLastSeqno       : 10992
appliedLatency         : 42.764
channels               : 1
clusterName            : alpha
currentEventId         : mysql-bin.000023:0000000505545003
currentTimeMillis      : 1389871897922
dataServerHost         : host1
extensions             :
host                   : host1
latestEpochNumber      : 0
masterConnectUri       : thl://localhost:/
masterListenUri        : thl://host1:2112/
```

```
maximumStoredSeqNo     : 10992
minimumStoredSeqNo     : 0
offlineRequests        : NONE
pendingError           : NONE
pendingErrorCode       : NONE
pendingErrorEventId    : NONE
pendingErrorSeqno      : -1
pendingExceptionMessage: NONE
pipelineSource         : jdbc:mysql:thin://host1:13306/
relativeLatency        : 158296.922
resourcePrecedence     : 99
rmiPort                : 10000
role                   : master
seqnoType              : java.lang.Long
serviceName            : alpha
serviceType            : local
simpleServiceName      : alpha
siteName               : default
sourceId               : host1
state                  : ONLINE
timeInStateSeconds     : 165845.474
transitioningTo        :
uptimeSeconds          : 165850.047
useSSLConnection       : false
version                : Tungsten Replicator 7.1.2 build 81
Finished status command...
```

When monitoring, the primary concernrs beyond identifying and copying with any errors is to monitor the applied latency. LArger numbers for applied latency generally indicate the the information is being written out to disk effectively. There are a number of strategies that should be checked:

- Confirm that the Hadoop environment is running effectively. Any delays to writing to HDFS will impact the replicator.
- Adjust the block commit parameters. Tuning the block commit levels should find the balance between frequent updates to achieve the required latency, and generating files of a suitable file sizes so that Hadoop can process them effectively for processing through map/reduce. You should try both increasing and reducing the sizes to find and figure out the the correct settings according to your source data.

## 4.6.4.5. Troubleshooting Hadoop Replication

Replicating to Hadoop involves a number of discrete, specific steps. Due to the batch and multi-stage nature of the extract and apply process, replication can stall or stop due to a variety of issues.

### 4.6.4.5.1. Errors Reading/Writing `commitseqno.0` File

During initial installation, or when starting up replication, the replicator may report that the `commitseqno.0` can not be created or written properly, or during startup, that the file cannot be read.

The following checks and recovery procedures can be tried:

- Check the permissions of the directory to the `commitseqno.0` file, the file itself, and the ownership:

```
shell> hadoop fs -ls -R /user/tungsten/metadata
drwxr-xr-x   - cloudera cloudera          0 2020-01-14 10:40 /user/tungsten/metadata/alpha
-rw-r--r--   3 cloudera cloudera        251 2020-01-14 10:40 /user/tungsten/metadata/alpha/commitseqno.0
```

- Check that the file is writable and is not empty. An empty file may indicate a problem updating the content with the new sequence number.
- Check the content of the file is correct. The content should be a JSON structure containing the replicator state and position information. For example:

```
shell> hadoop fs -cat /user/tungsten/metadata/alpha/commitseqno.0
{
  "appliedLatency" : "0",
  "epochNumber" : "0",
  "fragno" : "0",
  "shardId" : "dna",
  "seqno" : "8",
  "eventId" : "mysql-bin.000015:0000000000103156;0",
  "extractedTstamp" : "1578998421000"
  "lastFrag" : "true",
  "sourceId" : "host1"
}
```

- Try deleting the `commitseqno.0` file and placing the replicator online:

```
shell> hadoop fs -rm /user/tungsten/metadata/alpha/commitseqno.0
shell> trepctl online
```

#### 4.6.4.5.2. Recovering from Replication Failure

If the replication fails, is manually stopped, or the host needs to be restarted, replication should continue from the last point When replication was stopped. Files that were being written when replication was last running will be overwritten and the information recreated.

Unlike other Heterogeneous replication implementations, the Hadoop applier stores the current replication state and restart position in a file within the HDFS of the target Hadoop environment. To recover from failed replication, this file must be deleted, so that the THL can be re-read from the Source and CSV files will be recreated and applied into HDFS.

1. On the Applier, put the replicator offline:

   ```
   shell> trepctl offline
   ```

2. Remove the THL files from the Applier:

   ```
   shell> trepctl reset -thl
   ```

3. Remove the staging CSV files replicated into Hadoop:

   ```
   shell> hadoop fs -rm -r /user/tungsten/staging
   ```

4. Reset the restart position:

   ```
   shell> rm /opt/continuent/tungsten/tungsten-replicator/data/alpha/commitseqno.0
   ```

   Replace `alpha` and `/opt/continuent` with the corresponding service name and installation location.

5. Restart replication on the Applier; this will start to recreate the THL files from the MySQL binary log:

   ```
   shell> trepctl online
   ```

#### 4.6.4.5.3. Missing Primary Key

Replication may fail at the applier stage if the source data does not contain the correct ROW format and information, including the primary key data. trepctl may report the following error:

```
...
pendingErrorEventId    : mysql-bin.000015:0000000000143981;0
pendingErrorSeqno      : 10
pendingExceptionMessage: Wrapped com.continuent.tungsten.replicator.ReplicatorException: »
    Unable to find a primary key for dna.alt_allele_attrib and there is no default »
    from property stagePkeyColumn (../../tungsten-replicator//samples/scripts/batch/hdfs-merge.js#18)
pipelineSource         : UNKNOWN
relativeLatency        : -1.0
...
```

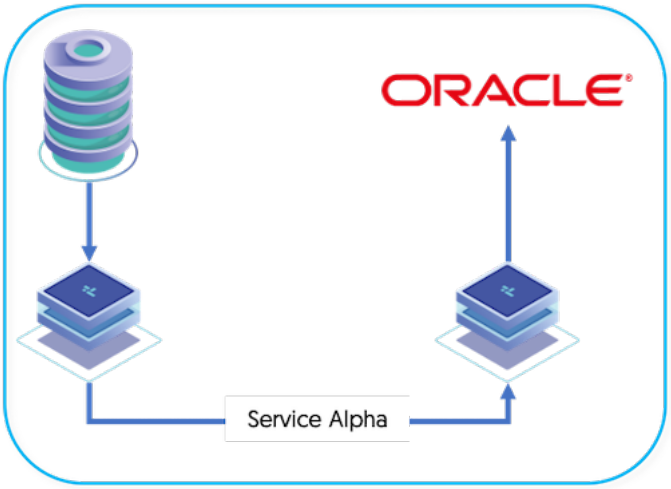
If the primary key was missing in the source data, the table structure on the source must be updated, and the THL information recreated.

## 4.7. Deploying the Oracle Applier

| Replication Operation Support | |
|---|---|
| Statements Replicated | No |
| Rows Replicated | Yes |
| Schema Replicated | No |
| ddlscan Supported | Yes |

Tungsten Cluster supports replication to Oracle as a datasource. This allows replication of data from MySQL to Oracle. See Section B.1.2, “Database Support” for more details.

Figure 4.9. Topologies: Replicating to Oracle




Replication in these configurations operates using two separate replicators:

- Replicator on the Extractor, extracts the information from the source database into THL.
- Replicator on the Applier reads the information from the remote replicator as THL, and applies that to the target database.

## 4.7.1. Preparing for Oracle Replication

Configure the source and target hosts following the prerequisites outlined in Appendix B, *Prerequisites* followed by the additional prerequisites specific to Oracle Targets outlined in Section 4.7.1.1, "Additional Prerequisites for Oracle Targets" then finally follow the appropriate steps for the required extractor topology outlined in Chapter 3, *Deploying MySQL Extractors*.

When replicating from MySQL to Oracle there are a number of datatype differences that should be accommodated to ensure reliable replication of the information. The core differences are described in Table 4.3, "Data Type differences when replicating data from MySQL to Oracle".

Table 4.3. Data Type differences when replicating data from MySQL to Oracle

| MySQL Datatype | Oracle Datatype | Notes |
|---|---|---|
| `INT` | `NUMBER(10, 0)` | |
| `BIGINT` | `NUMBER(19, 0)` | |
| `TINYINT` | `NUMBER(3, 0)` | |
| `SMALLINT` | `NUMBER(5, 0)` | |
| `MEDIUMINT` | `NUMBER(7, 0)` | |
| `DECIMAL(x,y)` | `NUMBER(x, y)` | |
| `FLOAT` | `FLOAT` | |
| `CHAR(n)` | `CHAR(n)` | |
| `VARCHAR(n)` | `VARCHAR2(n)` | For sizes less than 2000 bytes data can be replicated. For lengths larger than 2000 bytes, the data will be truncated when written into Oracle |
| `DATE` | `DATE` | |
| `DATETIME` | `DATE` | |
| `TIMESTAMP` | `DATE` | |
| `TEXT` | `CLOB` | Replicator can transform `TEXT` into `CLOB` or `VARCHAR(N)`. If you choose VARCHAR(N) on Oracle, the length of the data accepted by Oracle will be limited to 4000. This is limitation of Oracle. The size of `CLOB` columns within Oracle is calculated in terabytes. If `TEXT` fields on MySQL are known to be less than 4000 bytes (not characters) long, then `VARCHAR(4000)` can be used on Oracle. This may be faster than using `CLOB`. |

| MySQL Datatype | Oracle Datatype | Notes |
|---|---|---|
| `BLOB` | `BLOB` | |
| `ENUM(...)` | `VARCHAR(255)` | Use the `EnumToString` filter |
| `SET(...)` | `VARCHAR(255)` | Use the `SetToString` filter |

When replicating to Oracle, the ddlscan command can be used to generate DDL appropriate for the supported data types in the target database. In MySQL to Oracle deployments the DDL can be read from the MySQL server and generated for the Oracle server so that replication can begin without manually creating the Oracle specific DDL.

In addition, the following DDL differences and requirements exist:

- Column orders on MySQL and Oracle must match, but column names do not have to match.

  Using the `dropcolumn` filter, columns can be dropped and ignored if required.

- Each table within MySQL should have a Primary Key. Without a primary key, full-row based lookups are performed on the data when performing `UPDATE` or `DELETE` operations. With a primary key, the `pkey` filter can add metadata to the `UPDATE`/`DELETE` event, enabling faster application of events within Oracle.

- Indexes on MySQL and Oracle do not have to match. This allows for different index types and tuning between the two systems according to application and dataserver performance requirements.

- Keywords that are restricted on Oracle should not be used within MySQL as table, column or database names. For example, the keyword `SESSION` is not allowed within Oracle. Tungsten Cluster determines the column name from the target database metadata by position (column reference), not name, so replication will not fail, but applications may need to be adapted. For compatibility, try to avoid Oracle keywords.

For more information on differences between MySQL and Oracle, see Oracle and MySQL Compared.

To make the process of migration from MySQL to Oracle easier, Tungsten Cluster includes a tool called ddlscan which will read table definitions from MySQL and create appropriate Oracle table definitions to use during replication.

For reference information on the ddlscan tool, see Section 8.6, “The ddlscan Command”.

When replicating to Oracle there are a number of key steps that must be performed. The primary process is the preparation of the Oracle database and DDL for the database schema that are being replicated. Although DDL statements will be replicated to Oracle, they will often fail because of SQL language differences. Because of this, tables within Oracle must be created before replication starts.

### 4.7.1.1. Additional Prerequisites for Oracle Targets

When applying to oracle there are additional prerequisites required to ensure the replicator can connect to, and apply to, the target database

For remote Oracle targets (Offboard Applier)

To enable the replicator to apply to a remote Oracle Instance, the Replicator host will require an Oracle Client installation, with an appropriate TNS entry configured in the `tnsnames.ora` file

In addition, the environment for the tungsten OS user will need to be configured with `ORACLE_HOME` and `LD_LIBRARY_PATH` variables

For remote and local Oracle targets

Before installing you need to ensure that you have the `ojdbc7.jar` file in the correct location.

This can be copied to either:

- `$ORACLE_HOME/jdbc/lib`, or
- `/opt/continuent/software/tungsten-replicator-7.1.2-81/tungsten_replicator/lib`

### 4.7.1.2. Configure the Oracle database

Before installing replication, the Oracle target database must be configured:

- A user and schema must exist for each database from MySQL that you want to replicate. In addition, the schema used by the services within Tungsten Cluster must have an associated schema and user name.

  For example, if you are replicating the database `sales` to Oracle, the following statements must be executed to create a suitable schema. This can be performed through any connection, including sqlplus:

```
shell> sqlplus sys/oracle as sysdba
SQL> CREATE USER sales IDENTIFIED BY password DEFAULT TABLESPACE DEMO QUOTA UNLIMITED ON DEMO;
```

The above assumes a suitable tablespace has been created (`DEMO` in this case).

- A schema must also be created for each service replicating into Oracle. For example, if the service is called `alpha`, then the `tungsten_alpha` schema/user must be created. The same command can be used:

```
SQL> CREATE USER tungsten_alpha IDENTIFIED BY password DEFAULT TABLESPACE DEMO QUOTA UNLIMITED ON DEMO;
```

- One of the users used above must be configured so that it has the rights to connect to Oracle and has all rights so that it can execute statements on any schema:

```
SQL> GRANT CONNECT TO tungsten_alpha;
SQL> GRANT DBA TO tungsten_alpha;
```

The user/password combination selected will be required when configuring the Applier replication service.

### 4.7.1.3. Create the Destination Schema

On the host which has been already configured as the Extractor, use ddlscan to extract the DDL for Oracle:

```
shell> cd tungsten-replicator-7.1.2-81
shell> ./bin/ddlscan -user tungsten -url 'jdbc:mysql:thin://host1:3306/access_log' \
    -pass password -template ddl-mysql-oracle.vm -db access_log
```

The output should be captured and checked before applying it to your Oracle instance:

```
shell> ./bin/ddlscan -user tungsten -url 'jdbc:mysql:thin://host1:3306/access_log' \
    -pass password -template ddl-mysql-oracle.vm -db access_log > access_log.ddl
```

If you are happy with the output, it can be executed against your target Oracle database:

```
shell> cat access_log.ddl | sqlplus sys/oracle as sysdba
```

The generated DDL includes statements to drop existing tables if they exist. This will fail in a new installation, but the output can be ignored.

Once the process has been completed for this database, it must be repeated for each database that you plan on replicating from Oracle to MySQL.

## 4.7.2. Install Oracle Applier

The Applier replicator will read the THL from the remote Extractor and apply it into Oracle using a standard JDBC connection. The Applier replicator needs to know the Extractor hostname, and the datasource type.

1. Unpack the Tungsten Replicator distribution in staging directory:

```
shell> tar zxf tungsten-replicator-7.1.2-81.tar.gz
```

2. Change into the staging directory:

```
shell> cd tungsten-replicator-7.1.2-81
```

3. Obtain a copy of the Oracle JDBC driver and copy it into the `tungsten-replicator/lib` directory:

```
shell> cp ojdbc7.jar ./tungsten-replicator/lib/
```

4. Configure the installation using tpm:

   Show Staging

   Show INI

```
shell> ./tools/tpm configure defaults \
    --reset \
    --install-directory=/opt/continuent \
    --profile-script=~/.bash_profile \
    --skip-validation-check=InstallerMasterSlaveCheck \
    --rest-api-admin-user=apiuser \
    --rest-api-admin-pass=secret

shell> ./tools/tpm configure alpha \
    --master=sourcehost \
    --members=localhost \
    --datasource-type=oracle \
    --datasource-oracle-service=ORCL \
    --datasource-user=tungsten_alpha \
    --datasource-password=secret \
    --svc-applier-filters=dropstatementdata
```

```
shell> vi /etc/tungsten/tungsten.ini
```

```
[defaults]
install-directory=/opt/continuent
profile-script=~/.bash_profile
skip-validation-check=InstallerMasterSlaveCheck
rest-api-admin-user=apiuser
rest-api-admin-pass=secret

[alpha]
master=sourcehost
members=localhost
datasource-type=oracle
datasource-oracle-service=ORCL
datasource-user=tungsten_alpha
datasource-password=secret
svc-applier-filters=dropstatementdata
```

Configuration group **defaults**

The description of each of the options is shown below; click the icon to hide this detail:

Click the icon to show a detailed description of each argument.

- `--reset` [422]

  `reset` [422]

  For staging configurations, deletes all pre-existing configuration information between updating with the new configuration values.

- `--install-directory=/opt/continuent` [409]

  `install-directory=/opt/continuent` [409]

  Path to the directory where the active deployment will be installed. The configured directory will contain the software, THL and relay log information unless configured otherwise.

- `--profile-script=~/.bash_profile` [417]

  `profile-script=~/.bash_profile` [417]

  Append commands to include env.sh in this profile script

- `--skip-validation-check=InstallerMasterSlaveCheck` [369]

  `skip-validation-check=InstallerMasterSlaveCheck` [369]

  The `--skip-validation-check` [369] disables a given validation check. If any validation check fails, the installation, validation or configuration will automatically stop.

  > **Warning**
  >
  > Using this option enables you to bypass the specified check, although skipping a check may lead to an invalid or non-working configuration.

  You can identify a given check if an error or warning has been raised during configuration. For example, the default table type check:

  ```
  ...
  ERROR >> centos >> The datasource root@centos:3306 (WITH PASSWORD) »
   uses MyISAM as the default storage engine (MySQLDefaultTableTypeCheck)
  ...
  ```

  The check in this case is `MySQLDefaultTableTypeCheck` [380], and could be ignored using `--skip-validation-check=MySQLDefaultTableTypeCheck` [369].

  Setting both `--skip-validation-check` [369] and `--enable-validation-check` [367] is equivalent to explicitly disabling the specified check.

- `--rest-api-admin-user=apiuser` [422]

  `rest-api-admin-user=apiuser` [422]
  Optional: Must be specified along with rest-api-admin-pass if you wish to access the full API features and use the Dashboard GUI for cluster installations.

- `--rest-api-admin-pass=secret` [422]

  `rest-api-admin-pass=secret` [422]

Optional: Must be specified along with rest-api-admin-user if you wish to access the full API features.

Configuration group **alpha**

The description of each of the options is shown below; click the icon to hide this detail:

Click the icon to show a detailed description of each argument.

- `--master=sourcehost` [412]

  `master=sourcehost` [412]

  The hostname of the primary (extractor) within the current service.

- `--members=localhost` [413]

  `members=localhost` [413]

  Hostnames for the dataservice members

- `--datasource-type=oracle` [402]

  `datasource-type=oracle` [402]

  Database type

- `--datasource-oracle-service=ORCL` [402]

  `datasource-oracle-service=ORCL` [402]

  Oracle Service Name

- `--datasource-user=tungsten_alpha` [421]

  `datasource-user=tungsten_alpha` [421]

  For databases that required authentication, the username to use when connecting to the database using the corresponding connection method (native, JDBC, etc.).

- `--datasource-password=secret` [420]

  `datasource-password=secret` [420]

  The password to be used when connecting to the database using the corresponding `--replication-user` [421].

- `--svc-applier-filters=dropstatementdata` [424]

  `svc-applier-filters=dropstatementdata` [424]

  Replication service applier filters

`replication-host` [420] should be added to the above configuration if the target Oracle Database is on a different host to the applier installation

5. **Note**

   > If you plan to make full use of the REST API (which is enabled by default) you will need to also configure a username and password for API access. This must be done by specifying the following options in your configuration:
   >
   > ```
   > rest-api-admin-user=tungsten
   > rest-api-admin-pass=secret
   > ```

6. Once the prerequisites and configuring of the installation has been completed, the software can be installed:

```
shell> ./tools/tpm install
```

If the installation process fails, check the output of the `/tmp/tungsten-configure.log` file for more information about the root cause.

Once the installation has completed, the status of the service should be reported. The service should be online and reading events from the Extractor replicator.

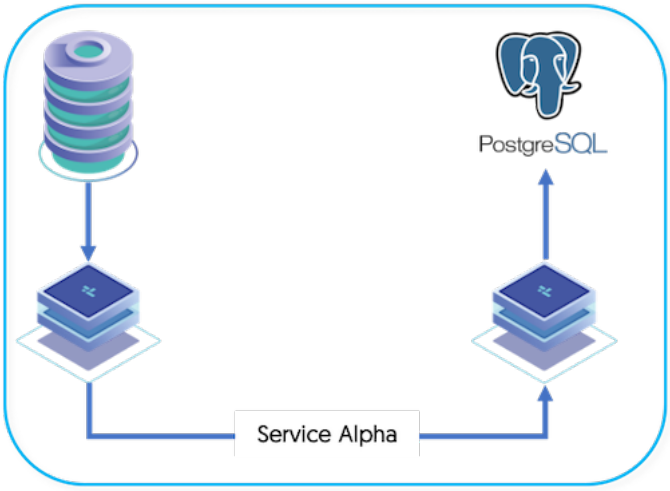
The status of the replicator can be checked and monitored by using the trepctl command.

# 4.8. Deploying the PostgreSQL Applier

Deployment of replication to PostgreSQL service operates as follows:

- Service Alpha on the Extractor, extracts the information from the MySQL binary log into THL.
- Service Alpha on the Applier reads the information from the remote replicator as THL, and applies that to PostgreSQL using a standard JDBC driver by constructing PostgreSQL compatible SQL to insert, update and delete the target data.

Figure 4.10. Topologies: Replicating to PostgreSQL




The two replication services can operate on the same machine, (See Section 5.3, "Deploying Multiple Replicators on a Single Host") or they can be installed on two different machines.

## 4.8.1. Preparing for PostgreSQL Replication

Configure the source and target hosts following the prerequisites outlined in Appendix B, *Prerequisites* then follow the appropriate steps for the required extractor topology outlined in Chapter 3, *Deploying MySQL Extractors*.

### 4.8.1.1. PostgreSQL Database Setup

For replication to PostgreSQL hosts, you must ensure that the networking and user configuration has been configured correctly.

#### 4.8.1.1.1. PostgreSQL Version Support

| Database | Version | Support Status | Notes |
|---|---|---|---|
| PostgreSQL | 9.5, 9.6 | Primary platform (applier only) | |

#### 4.8.1.1.2. Enable PostgreSQL Networking

Within the PostgreSQL configuration, two changes need to be made:

- Configure the networking so that the listen address for the PostgreSQL server is configured correctly for this edit. Edit the `/etc/postgresql/main/postgresql.conf` file and edit the `listen_address` line either to `*` or to an explicit IP address. For example:

```
listen_addresses = '192.168.3.73'
```

- Edit the `/etc/postgresql/main/pg_hba.conf` file and ensure that the password properties match the password settings and hostname limitations. In particular, the replicator will communicate over the public IP address, not localhost, and so you must ensure that network-based connections using a user/password combination are allowed. For example, you may want to add a line to the file that provides network-wide access, or at least access for the local network range:

```
local   all             all                                     md5
```

#### 4.8.1.1.3. User Configuration

A suitable user must be created with rights and permissions to create databases, as this is required by the replicator to create databases, tables, and other objects. The creatuser command can be used for this purpose. The `--createdb` adds the `CREATEDB` permission:

```
shell> createuser tungsten --createdb
```

You will be prompted to provide a password for the user.

Alternatively, you can create the user and permissions through the psql interface:

```
shell> sudo -u postgres psql --port=5433 --user=postgres postgres
    Type "help" for help.

    postgres=# CREATE ROLE tungsten WITH LOGIN PASSWORD 'password';
    postgres=# ALTER ROLE tungsten CREATEDB;
```

You may also want to grant specific privileges to existing databases which must be done within the psql interface:

```
shell> sudo -u postgres psql --port=5433 --user=postgres postgres
    Type "help" for help.

    postgres=# GRANT ALL ON DATABASE postgres TO tungsten;
```

## 4.8.2. Install PostgreSQL Applier

Once you have completed the configuration of the PostgreSQL database, you can configure and install the PostgreSQL applier as described using the steps below.

1. Unpack the Tungsten Replicator distribution in staging directory:

   ```
   shell> tar zxf tungsten-replicator-7.1.2-81.tar.gz
   ```

2. Change into the staging directory:

   ```
   shell> cd tungsten-replicator-7.1.2-81
   ```

3. Configure the installation using tpm:

   Show Staging

   Show INI

   ```
   shell> ./tools/tpm configure defaults \
       --reset \
       --install-directory=/opt/continuent \
       --user=tungsten \
       --profile-script=~/.bash_profile \
       --rest-api-admin-user=apiuser \
       --rest-api-admin-pass=secret

   shell> ./tools/tpm configure alpha \
       --master=sourcehost \
       --members=localhost,sourcehost \
       --datasource-type=postgresql \
       --postgresql-dbname=dbname \
       --replication-user=tungsten \
       --replication-password=secret \
       --replication-host=remotedbhost \
       --replication-port=5432
   ```

   ```
   shell> vi /etc/tungsten/tungsten.ini
   ```

   ```
   [defaults]
   install-directory=/opt/continuent
   user=tungsten
   profile-script=~/.bash_profile
   rest-api-admin-user=apiuser
   rest-api-admin-pass=secret

   [alpha]
   master=sourcehost
   members=localhost,sourcehost
   datasource-type=postgresql
   postgresql-dbname=dbname
   replication-user=tungsten
   replication-password=secret
   replication-host=remotedbhost
   replication-port=5432
   ```

Configuration group **defaults**

The description of each of the options is shown below; click the icon to hide this detail:

Click the icon to show a detailed description of each argument.

- `--reset` [422]

  `reset` [422]

  For staging configurations, deletes all pre-existing configuration information between updating with the new configuration values.

- `--install-directory=/opt/continuent` [409]

  `install-directory=/opt/continuent` [409]

  Path to the directory where the active deployment will be installed. The configured directory will contain the software, THL and relay log information unless configured otherwise.

- `--user=tungsten` [429]

  `user=tungsten` [429]

  System User

- `--profile-script=~/.bash_profile` [417]

  `profile-script=~/.bash_profile` [417]

  Append commands to include env.sh in this profile script

- `--rest-api-admin-user=apiuser` [422]

  `rest-api-admin-user=apiuser` [422]
  Optional: Must be specified along with rest-api-admin-pass if you wish to access the full API features and use the Dashboard GUI for cluster installations.

- `--rest-api-admin-pass=secret` [422]

  `rest-api-admin-pass=secret` [422]
  Optional: Must be specified along with rest-api-admin-user if you wish to access the full API features.

Configuration group **alpha**

The description of each of the options is shown below; click the icon to hide this detail:

Click the icon to show a detailed description of each argument.

- `--master=sourcehost` [412]

  `master=sourcehost` [412]

  The hostname of the primary (extractor) within the current service.

- `--members=localhost,sourcehost` [413]

  `members=localhost,sourcehost` [413]

  Hostnames for the dataservice members

- `--datasource-type=postgresql` [402]

  `datasource-type=postgresql` [402]

  Database type

- `--postgresql-dbname=dbname` [416]

  `postgresql-dbname=dbname` [416]

  Name of the database to replicate

- `--replication-user=tungsten` [421]

  `replication-user=tungsten` [421]

  For databases that required authentication, the username to use when connecting to the database using the corresponding connection method (native, JDBC, etc.).

- `--replication-password=secret` [420]

  `replication-password=secret` [420]

  The password to be used when connecting to the database using the corresponding `--replication-user` [421].

- `--replication-host=remotedbhost` [420]

  `replication-host=remotedbhost` [420]

  Hostname of the datasource where the database is located. If the specified hostname matches the current host or member name, the database is assumed to be local. If the hostnames do not match, extraction is assumed to be via remote access. For MySQL hosts, this configures a remote replication Replica (relay) connection.

- `--replication-port=5432` [421]

  `replication-port=5432` [421]

  The network port used to connect to the database server. The default port used depends on the database being configured.

4. **Note**

   > If you plan to make full use of the REST API (which is enabled by default) you will need to also configure a username and password for API access. This must be done by specifying the following options in your configuration:
   >
   > ```
   > rest-api-admin-user=tungsten
   > rest-api-admin-pass=secret
   > ```

5. Once the prerequisites and configuring of the installation has been completed, the software can be installed:

```
shell> ./tools/tpm install
```

If the installation process fails, check the output of the `/tmp/tungsten-configure.log` file for more information about the root cause.

Once the replicators have started, the status of the service can be checked using trepctl. See Section 4.8.3, “Management and Monitoring of PostgreSQL Deployments” for more information.

## 4.8.3. Management and Monitoring of PostgreSQL Deployments

Once the two services — extractor and applier — have been installed, the services can be monitored using trepctl. To monitor the extractor service:

```
shell> trepctl status
Processing status command...
NAME                     VALUE
----                     -----
appliedLastEventId     : mysql-bin.000008:0000000000412301;0
appliedLastSeqno       : 1296
appliedLatency         : 1.889
channels               : 1
clusterName            : epsilon
currentEventId         : mysql-bin.000008:0000000000412301
currentTimeMillis      : 1377097812795
dataServerHost         : host1
extensions             :
latestEpochNumber      : 1286
masterConnectUri       : thl://localhost:/
masterListenUri        : thl://host2:2112/
maximumStoredSeqNo     : 1296
minimumStoredSeqNo     : 0
offlineRequests        : NONE
pendingError           : NONE
pendingErrorCode       : NONE
pendingErrorEventId    : NONE
pendingErrorSeqno      : -1
pendingExceptionMessage: NONE
pipelineSource         : jdbc:mysql:thin://host1:13306/
```

```
relativeLatency        : 177444.795
resourcePrecedence     : 99
rmiPort                : 10000
role                   : master
seqnoType              : java.lang.Long
serviceName            : alpha
serviceType            : local
simpleServiceName      : alpha
siteName               : default
sourceId               : host1
state                  : ONLINE
timeInStateSeconds     : 177443.948
transitioningTo        :
uptimeSeconds          : 177461.483
version                : Tungsten Replicator 7.1.2 build 81
Finished status command...
```

The replicator service operates just the same as a standard Extractor service of a typical MySQL replication service.

The PostgreSQL applier service can be accessed either remotely from the Extractor:

```
shell> trepctl -host host2 status
...
```

Or locally on the Applier host:

```
shell> trepctl status
Processing status command...
NAME                     VALUE
----                     -----
appliedLastEventId     : mysql-bin.000008:0000000000412301;0
appliedLastSeqno       : 1296
appliedLatency         : 10.253
channels               : 1
clusterName            : alpha
currentEventId         : NONE
currentTimeMillis      : 1377098139212
dataServerHost         : host2
extensions             :
latestEpochNumber      : 1286
masterConnectUri       : thl://host1:2112/
masterListenUri        : null
maximumStoredSeqNo     : 1296
minimumStoredSeqNo     : 0
offlineRequests        : NONE
pendingError           : NONE
pendingErrorCode       : NONE
pendingErrorEventId    : NONE
pendingErrorSeqno      : -1
pendingExceptionMessage: NONE
pipelineSource         : thl://host1:2112/
relativeLatency        : 177771.212
resourcePrecedence     : 99
rmiPort                : 10000
role                   : slave
seqnoType              : java.lang.Long
serviceName            : alpha
serviceType            : local
simpleServiceName      : alpha
siteName               : default
sourceId               : host2
state                  : ONLINE
timeInStateSeconds     : 177783.343
transitioningTo        :
uptimeSeconds          : 180631.276
version                : Tungsten Replicator 7.1.2 build 81
Finished status command...
```

Monitoring the status of replication between the Source and Target is also the same. The `appliedLastSeqno` still indicates the sequence number that has been applied to PostgreSQL, and the event ID from PostgreSQL can still be identified from `appliedLastEventId`.

Sequence numbers between the two hosts should match, as in a Primary/Replica deployment, but due to the method used to replicate, the applied latency may be higher. Tables that do not use primary keys, or large individual row updates may cause increased latency differences.

## 4.9. Deploying the Amazon S3 CSV Applier

Amazon S3 is a cloud-based data storage service that integrates with other Amazon services. Replication for Amazon S3 moves data from MySQL datastores, in real-time to csv files stored within an S3 bucket.

Replication to Amazon S3 operates as follows:

- Data is extracted from the source database into THL.
- When extracting the data from the THL, the Amazon S3 replicator writes the data into CSV files according to the name of the source tables. The files contain all of the row-based data, including the global transaction ID generated by the extractor during replication, and the operation type (insert, delete, etc) as part of the CSV data.
- The generated CSV files are loaded into Amazon S3 using either the s3cmd command or the aws s3 cli tools. This enables easy access to your Amazon S3 installation and simplifies the loading.

Setting up replication requires setting up both the Extractor and Applier components as two different configurations, one for MySQL and the other for Amazon S3. Replication also requires some additional steps to ensure that S3 is ready to accept the replicated data that has been extracted. Tungsten Replicator provides all the tools required to perform these operations during the installation and setup.

## 4.9.1. S3 Replication Operation

The S3 applier makes use of the JavaScript based batch loading system (see Section 5.6.4, "JavaScript Batchloader Scripts"). This constructs change data from the source-database. The change data is then written into csv files, optional compressed, and loaded into an S3 bucket.

The full replication of information operates as follows:

1. Data is extracted from the source database using the standard extractor, for example by reading the row change data from the binlog in MySQL.
2. The Section 11.4.5, "ColumnName Filter" filter is used to extract column name information from the database. This enables the row-change information to be tagged with the corresponding column information. The data changes, and corresponding row names, are stored in the THL.

   The Section 11.4.31, "PrimaryKey Filter" filter is used to extract primary key data from the source tables.
3. On the Applier replicator, the THL data is read and written into batch-files in the character-separated value format.

   The information in these files is change data, and contains not only the original row values from the source tables, but also metadata about the operation performed (i.e. `INSERT`, `DELETE` or `UPDATE`, and the primary key of for each table. All `UPDATE` statements are recorded as a `DELETE` of the existing data, and an `INSERT` of the new data.

   In addition to these core operation types, the batch applier can also be configured to record `UPDATE` operations that result in `INSERT` or `DELETE` rows.

The staging files created by the replicator are in a specific format that incorporates change and operation information in addition to the original row data.

- The format of the files is a character separated values file, with each row separated by a newline, and individual fields separated by the character `0x01`.
- The content of the file consists of the full row data extracted from the Source, plus metadata describing the operation for each row, the sequence number, and then the full row information.

| Operation | Sequence No | Table-specific primary key | DateTime | Table-columns... |
|---|---|---|---|---|
| OPTYPE | *SEQNO* [549] that generated this row | PRIMARYKEY | DATATIME of source table commit | |

The operation field will match one of the following values

| Operation | Description | Notes |
|---|---|---|
| I | Row is an `INSERT` of new data | |
| D | Row is `DELETE` of existing data | |
| UI | Row is an `UPDATE` which caused `INSERT` of data | |
| UD | Row is an `UPDATE` which caused `DELETE` of data | |

For example, the MySQL row from an `INSERT` of:

```
|  3 | #1 Single | 2006 | Cats and Dogs (#1.4)         |
```

Is represented within the CSV files generated as:

```
"I","5","3","2014-07-31 14:29:17.000","3","#1 Single","2006","Cats and Dogs (#1.4)"
```

The character separator, and whether to use quoting, are configurable within the replicator when it is deployed. For S3, the default behavior is to generate quoted and comma separated fields.

As the target for the Amazon S3 Applier is not a relational database in the sense of traditional Tungsten replication, the replicator stores it's apply position as a JSON structure on the local filesystem.

This allows the replicator to know it's starting position in the case of a restart.

The file is located in the following directory: `/opt/continuent/metadata/applier/serviceName` and is called `commitseqno.0`

The contents of the file will look something like the following, and should NOT be edited unless advised to do so by Continuent Support

```
{
  "sourceId" : "ext01",
  "epochNumber" : "0",
  "fragno" : "0",
  "eventId" : "mysql-bin.000002:0000000000134613;27",
  "seqno" : "427",
  "lastFrag" : "true",
  "extractedTstamp" : "1687439308000",
  "appliedLatency" : "0",
  "shardId" : "demo"
}
```

## 4.9.2. Preparing for Amazon S3 Replication

Preparing the hosts for the replication process requires setting some key configuration parameters within the MySQL server to ensure that data is stored and written correctly.

Configure the source and target hosts following the prerequisites outlined in Appendix B, *Prerequisites* then follow the appropriate steps for the required extractor topology outlined in Chapter 3, *Deploying MySQL Extractors*.

The following are *required* for replication to Amazon S3:

- An existing Amazon Web Services (AWS) account, and either the AWS Access Key and Secret Key, or configured IAM Roles, required to interact with the account through the API. For information on creating IAM Roles, see Section 4.2.2.2, "Configuring Identity Access Management within AWS"
- A configured Amazon S3 service. If the S3 service has not already been configured, visit the AWS console and sign up for the Amazon S3 service.
- If using the s3cmd, you should then configure the command to automatically connect to the Amazon S3 service without requiring further authentication, the `.s3cfg` in the `tungsten` users home directory should be configured as follows:
  - Using Access Keys:

    ```
    [default]
    access_key = ACCESS_KEY
    secret_key = SECRET_KEY
    ```

  - Using IAM Roles: Leave values blank - copy example as is

    ```
    [default]
    access_key =
    secret_key =
    security_token =
    ```

- Create an S3 bucket that will be used to hold the CSV files that are generated by the replicator. This can be achieved either through the web interface, or via the command-line, for example:

  ```
  shell> s3cmd mb s3://tungsten-csv
  ```

- Create an `s3-config-servicename.json` file based on the sample provided within `cluster-home/samples/conf/s3-config-servicename.json` within the Tungsten Replicator staging directory, or using the example below.

  Once created, the file will be copied into the `/opt/continuent/share` directory to be used by the batch applier script.

  If multiple services are being created, one file must be created for each service.

  The following example shows the use of Access and Secret Keys:

  ```
  {
    "awsS3Path" : "s3://your-bucket-for-s3/s3-test",
    "awsAccessKey" : "access-key-id",
    "awsSecretKey" : "secret-access-key",
    "cleanUpS3Files" : "true"
  }
  ```

  The following example shows the use of IAM Roles:

```
{
  "awsS3Path" : "s3://your-bucket-for-s3/s3-test",
  "awsIAMRole" : "arn:iam-role",
}
```

The allowed options for this file are as follows:

- `awsS3Path` — the location within your S3 storage where files should be loaded.
- `awsAccessKey` — the S3 access key to access your S3 storage. Not required if awsIAMRole is used.
- `awsSecretKey` — the S3 secret key associated with the Access Key. Not required if awsIAMRole is used.
- `awsIAMRole` — the IAM role configured to allow Redshift to interact with S3. Not required if awsAccessKey and awsSecretKey are in use.
- `s3Binary` — the binary to use for loading csv file up to S3. (Valid Values: s3cmd, s4cmd, aws) (Default: s3cmd)
- `gzipS3Files` — setting to true will result in the csv files being gzipped prior to loading into S3 (Default: false)

## 4.9.3. Install Amazon S3 Applier

Replication into S3 requires two separate replicator installations, one that extracts information from the source database, and a second that generates the CSV files, loads those files into S3.

The two replication services can operate on the same machine, (See Section 5.3, "Deploying Multiple Replicators on a Single Host") or they can be installed on two different machines.

Once you have completed the configuration of the Amazon S3 bucket, you can configure and install the applier as described using the steps below.

1. Before installing the applier, the following additions need adding to the extractor configuration. Apply the following parameter to the extractor configuration before installing the applier

   Add the following the `/etc/tungsten/tungsten.ini`

   ```
   [alpha]
   ...Existing Replicator Config...
   enable-heterogeneous-service=true

   shell> tpm update
   ```

   > **Note**
   >
   > The above step is only applicable for standalone extractors. If you are configuring replications from an existing Tungsten Cluster (Cluster-Extractor), follow the steps outlined here to ensure the cluster is configured correctly: Section 3.4.1, "Prepare: Replicating Data Out of a Cluster"

2. The applier can now be configured. Unpack the Tungsten Replicator distribution in staging directory:

   ```
   shell> tar zxf tungsten-replicator-7.1.2-81.tar.gz
   ```

3. Change into the staging directory:

   ```
   shell> cd tungsten-replicator-7.1.2-81
   ```

4. Configure the installation using tpm:

   Show Staging

   Show INI

   ```
   shell> ./tools/tpm configure defaults \
       --reset \
       --user=tungsten \
       --install-directory=/opt/continuent \
       --profile-script=~/.bash_profile \
       --rest-api-admin-user=apiuser \
       --rest-api-admin-pass=secret \
       --replicator-rest-api-ssl=true \
       --replicator-rest-api-port=8097 \
       --replicator-rest-api-authentication=true \
       --replicator-rest-api-address=0.0.0.0

   shell> ./tools/tpm configure alpha \
       --master=sourcehost \
       --members=localhost \
   ```

```
    --role=slave \
    --batch-enabled=true \
    --batch-load-template=s3 \
    --datasource-type=file \
    --enable-heterogeneous-service=true
```

```
shell> vi /etc/tungsten/tungsten.ini
```

```
[defaults]
user=tungsten
install-directory=/opt/continuent
profile-script=~/.bash_profile
rest-api-admin-user=apiuser
rest-api-admin-pass=secret
replicator-rest-api-ssl=true
replicator-rest-api-port=8097
replicator-rest-api-authentication=true
replicator-rest-api-address=0.0.0.0

[alpha]
master=sourcehost
members=localhost
role=slave
batch-enabled=true
batch-load-template=s3
datasource-type=file
enable-heterogeneous-service=true
```

Configuration group **`defaults`**

The description of each of the options is shown below; click the icon to hide this detail:

Click the icon to show a detailed description of each argument.

- `--reset` [422]

  `reset` [422]

  For staging configurations, deletes all pre-existing configuration information between updating with the new configuration values.

- `--user=tungsten` [429]

  `user=tungsten` [429]

  System User

- `--install-directory=/opt/continuent` [409]

  `install-directory=/opt/continuent` [409]

  Path to the directory where the active deployment will be installed. The configured directory will contain the software, THL and relay log information unless configured otherwise.

- `--profile-script=~/.bash_profile` [417]

  `profile-script=~/.bash_profile` [417]

  Append commands to include env.sh in this profile script

- `--rest-api-admin-user=apiuser` [422]

  `rest-api-admin-user=apiuser` [422]
  Optional: Must be specified along with rest-api-admin-pass if you wish to access the full API features and use the Dashboard GUI for cluster installations.

- `--rest-api-admin-pass=secret` [422]

  `rest-api-admin-pass=secret` [422]
  Optional: Must be specified along with rest-api-admin-user if you wish to access the full API features.

- `--replicator-rest-api-ssl=true` [421]

  `replicator-rest-api-ssl=true` [421]

  Enable SSL for the API.

- `--replicator-rest-api-port=8097`

  `replicator-rest-api-port=8097`

  Port for the Replicator API.

- `--replicator-rest-api-authentication=true` [421]

  `replicator-rest-api-authentication=true` [421]

  Enforce authentication for the API.

- `--replicator-rest-api-address=0.0.0.0` [421]

  `replicator-rest-api-address=0.0.0.0` [421]

  Address for the API to bind too.

Configuration group **`alpha`**

The description of each of the options is shown below; click the icon to hide this detail:

Click the icon to show a detailed description of each argument.

- `--master=sourcehost` [412]

  `master=sourcehost` [412]

  The hostname of the primary (extractor) within the current service.

- `--members=localhost` [413]

  `members=localhost` [413]

  Hostnames for the dataservice members

- `--role=slave` [422]

  `role=slave` [422]

  What is the replication role for this service?

- `--batch-enabled=true` [397]

  `batch-enabled=true` [397]

  Should the replicator service use a batch applier

- `--batch-load-template=s3` [398]

  `batch-load-template=s3` [398]

  Value for the loadBatchTemplate property

- `--datasource-type=file` [402]

  `datasource-type=file` [402]

  Database type

- `--enable-heterogeneous-service=true` [406]

  `enable-heterogeneous-service=true` [406]

  - On a Primary

    - `--mysql-use-bytes-for-string` [414] is set to false.

    - `colnames` filter is enabled (in the `binlog-to-q` stage to add column names to the THL information.

    - `pkey` filter is enabled (in the `binlog-to-q` and `q-to-dbms` stage), with the *`addPkeyToInserts`* and *`addColumnsToDeletes`* filter options set to false.

    - `enumtostring` filter is enabled (in the `q-to-thl` stage), to translate `ENUM` values to their string equivalents.
    - `settostring` filter is enabled (in the `q-to-thl` stage), to translate `SET` values to their string equivalents.
  - On a Replica
    - `--mysql-use-bytes-for-string` [414] is set to true.
    - `pkey` filter is enabled (`q-to-dbms` stage).

5. If your MySQL source is a Tungsten Cluster, ensure the additional steps below are also included in your applier configuration

   First, prepare the required filter configuration file as follows on the S3 applier host(s) only:

```
shell> mkdir -p /opt/continuent/share/
shell> cp tungsten-replicator/support/filters-config/convertstringfrommysql.json /opt/continuent/share/
```

   Then, include the following parameters in the configuration

```
property=replicator.stage.remote-to-thl.filters=convertstringfrommysql
property=replicator.filter.convertstringfrommysql.definitionsFile=/opt/continuent/share/convertstringfrommysql.json
```

6. 

   **Note**

   If you plan to make full use of the REST API (which is enabled by default) you will need to also configure a username and password for API access. This must be done by specifying the following options in your configuration:

```
rest-api-admin-user=tungsten
rest-api-admin-pass=secret
```

7. Once the prerequisites and configuring of the installation has been completed, the software can be installed:

```
shell> ./tools/tpm install
```

If the installation process fails, check the output of the `/tmp/tungsten-configure.log` file for more information about the root cause.

On the host that is loading data into S3, create the `s3-config-servicename.json` file and then copy that file into the `share` directory within the installed directory on that host. For example:

```
shell> cp s3-config-servicename.json /opt/continuent/share/
```

Now the services can be started:

```
shell> replicator start
```

# Chapter 5. Deployment: Advanced

## 5.1. Deploying the Replicator using the AWS Marketplace AMI

If you have an AWS account, you can take advantage of pre-built EC2 hosts, complete with all necessary pre-requisites in place, launched from an AWS Marketplace AMI.

Upon launch, a wizard will start and prompt you for a number of credentials to build a default configuration for Tungsten Replicator

For a complete end-to-end Replication Pipeline you will need:

- One host launched from the Tungsten Replicator for MySQL Source Extraction AMI for each Source you wish to extract from
- One or more hosts launched from the appropriate Target AMI to match your requirements

Source Databases

The Tungsten Replicator for MySQL Source Extraction is required for extraction from any of the following:

- MySQL hosted on another EC2 instance
- MySQL hosted on the same EC2 host launched from the AMI
- An existing Tungsten Clustering Installation
- Amazon RDS
- Amazon Aurora
- MySQL hosted on a remote non-AWS host
- Google Cloud SQL
- Microsoft Azure

Target Databases

- MySQL Targets include all of the following:
  - MySQL hosted on another EC2 instance
  - MySQL hosted on the same EC2 host launched from the AMI
  - An existing Tungsten Clustering Installation
  - Amazon RDS
  - Amazon Aurora
  - MySQL hosted on a remote non-AWS host
  - Google Cloud SQL
  - Microsoft Azure
- Amazon Redshift
- HP Vertica
- PostgreSQL (Including RDS)
- Apache Kafka
- Hadoop
- Oracle (Including RDS)
- Clickhouse
- MongoDB

### Note

Upon launch, the AMI does NOT include the required binaries for a locally hosted database instance. For a local install for either the extractor or the applier, this will need to be configured manually beforehand.

### Note

> If you plan to extract from an existing Tungsten Cluster (Cluster-Extractor) a number of changes may need to be applied to your cluster configuration, in addition your cluster must be running the same release as Tungsten Replicator. For more details on Cluster requirements consult the appropriate Applier specific pages here: Chapter 4, *Deploying Appliers*

### Note

> For any non-AWS hosted instances, ensure the appropriate inbound and outbound security rules are in place to allow WAN Communication.

## 5.1.1. Prepare Source/Target database instances

When using the AMI to configure an Extractor or Applier, it is important to ensure all the necessary target/source database pre-requisites are in place.

- For extraction, ensure your source MySQL Instance is configured as per the Database specific notes in Section B.4, “MySQL Database Setup”
- In addition, for Amazon based extraction, pay particular attention to Section B.4.6, “MySQL Unprivileged Users”
- For preparing the target database, specific notes for target pre-requisities, where appropriate, are detailed within each applier deployment section found at Chapter 4, *Deploying Appliers*
- Once you have prepared your sources and targets, you can now launch the relevant AMI's from the Marketplace
- Within your AWS Dashboard, you can find the AMI by searching within the Marketplace for "Continuent"
- Select the Extractor AMI and the Target AMI based on your choice of target database. Each AMI is restricted to only configure an applier based on the choice of target. There are no restrictions on extraction, providing the necessary pre-requisities are in place.
- Ensure you select a Security group that allows communication to the source and target databases, the require network ports are detailed in Section B.3.2.1, “Network Ports”

## 5.1.2. Launch and Configure AMI

After launching the AMI, obtain the public IP and connect to the shell using your preferred Terminal application, eg

```
shell> ssh -i your-key.pem ec2-user@publicIP
```

Upon connecting, you will see a welcome message, from here you can now connect as the tungsten user

```
shell> sudo su - tungsten
```

The launch wizard will start automatically and start prompting you for details regarding your source or target database.

It is advisable to configure the Extractor AMI first as you will need to provide details of the extractor when you configure the applier.

Once you have provided all the information to the wizard, you will be prompted on screen for the next steps.

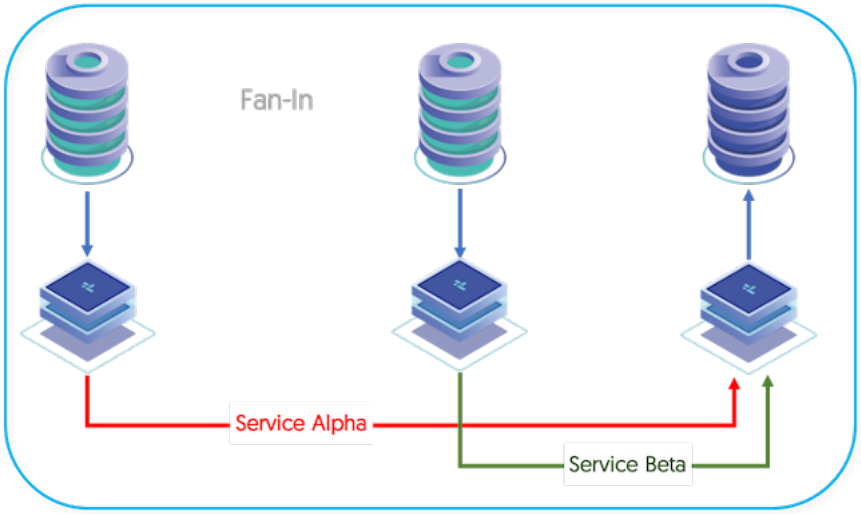
- In summary, the wizard will have completed the following:
  - Created `tungsten.ini` within /etc/tungsten
  - Created additional directories for software installation
  - Created additional configuration files depening upon target requirements
  - Created a log file of the Wizard execution within /home/tungsten/ami-launch/log
- The latest version of Tungsten Replicator will be unpacked within /opt/continuent/software
- The wizard does not install the software, this allows you to fine tune the configuration to suit your needs, such as adding additional filters, or adjusting memory and buffer allocations.
- For more information on all the possible configuration parameters, see Section 9.8, “tpm Configuration Options”
- You can now install the software, follow the on screen instructions displayed after Wizard completetion to install using tpm, or review Section 9.4.2, “Installation with INI File”
- For further reading and understanding of how to manage the replicator, review Chapter 7, *Operations Guide*
- For steps on starting and stopping the replicator, review Section 2.4, “Starting and Stopping Tungsten Replicator”

- For details on how to monitor and interact with the running replicator using the trepctl tool, review Section 8.20, "The trepctl Command"

## 5.2. Deploying a Fan-In Topology

The fan-in topology is the logical opposite of a Primary/Replica topology. In a fan-in topology, the data from two Sources is combined together on one Target. Fan-in topologies are often in situations where you have satellite databases, maybe for sales or retail operations, and need to combine that information together in a single database for processing.

Figure 5.1. Topologies: Fan-in




Some additional considerations need to be made when using fan-in topologies:

- If the same tables from each each machine are being merged together, it is possible to get collisions in the data where auto increment is used. The effects can be minimized by using increment offsets within the MySQL configuration:

```
auto-increment-offset = 1
auto-increment-increment = 4
```

- Fan-in can work more effectively, and be less prone to problems with the corresponding data by configuring specific tables at different sites. For example, with two sites in New York and San Jose databases and tables can be prefixed with the site name, i.e. `sjc_sales` and `nyc_sales`.

  Alternatively, a filter can be configured to rename the database `sales` dynamically to the corresponding location based tables. See Section 11.4.33, "Rename Filter" for more information.

- Statement-based replication will work for most instances, but where your statements are updating data dynamically within the statement, in fan-in the information may get increased according to the name of fan-in Sources. Update your configuration file to explicitly use row-based replication by adding the following to your `my.cnf` file:

```
binlog-format = row
```

- Triggers can cause problems during fan-in replication if two different statements from each Source and replicated to the Target and cause the operations to be triggered multiple times. Tungsten Replicator cannot prevent triggers from executing on the concentrator host and there is no way to selectively disable triggers. Check at the trigger level whether you are executing on a Source or Target. For more information, see Section C.4.1, "Triggers".

To create the configuration the Extractors and services must be specified, the topology specification takes care of the actual configuration:

Show Staging

Show INI

```
shell> ./tools/tpm configure epsilon \
    --topology=fan-in \
    --install-directory=/opt/continuent \
    --replication-user=tungsten \
    --replication-password=password \
    --master=host1,host2 \
    --members=host1,host2,host3 \
    --master-services=alpha,beta \
    --rest-api-admin-user=apiuser \
    --rest-api-admin-pass=secret
```

```
shell> vi /etc/tungsten/tungsten.ini
```

```
[epsilon]
topology=fan-in
install-directory=/opt/continuent
replication-user=tungsten
replication-password=password
master=host1,host2
members=host1,host2,host3
master-services=alpha,beta
rest-api-admin-user=apiuser
rest-api-admin-pass=secret
```

Configuration group **epsilon**

The description of each of the options is shown below; click the icon to hide this detail:

Click the icon to show a detailed description of each argument.

- `--topology=fan-in` [429]

  `topology=fan-in` [429]

  Replication topology for the dataservice.

- `--install-directory=/opt/continuent` [409]

  `install-directory=/opt/continuent` [409]

  Path to the directory where the active deployment will be installed. The configured directory will contain the software, THL and relay log information unless configured otherwise.

- `--replication-user=tungsten` [421]

  `replication-user=tungsten` [421]

  For databases that required authentication, the username to use when connecting to the database using the corresponding connection method (native, JDBC, etc.).

- `--replication-password=password` [420]

  `replication-password=password` [420]

  The password to be used when connecting to the database using the corresponding `--replication-user` [421].

- `--master=host1,host2` [412]

  `master=host1,host2` [412]

  The hostname of the primary (extractor) within the current service.

- `--members=host1,host2,host3` [413]

  `members=host1,host2,host3` [413]

  Hostnames for the dataservice members

- `--master-services=alpha,beta` [412]

  `master-services=alpha,beta` [412]

  Data service names that should be used on each Primary

- `--rest-api-admin-user=apiuser` [422]

  `rest-api-admin-user=apiuser` [422]
  Optional: Must be specified along with rest-api-admin-pass if you wish to access the full API features and use the Dashboard GUI for cluster installations.

- `--rest-api-admin-pass=secret` [422]

  `rest-api-admin-pass=secret` [422]
  Optional: Must be specified along with rest-api-admin-user if you wish to access the full API features.

For additional options supported for configuration with tpm, see Chapter 9, *The tpm Deployment Command*.

If the installation process fails, check the output of the `/tmp/tungsten-configure.log` file for more information about the root cause.

Once the installation has been completed, the service will be started and ready to use.

## 5.2.1. Management and Monitoring Fan-in Deployments

Once the service has been started, a quick view of the service status can be determined using trepctl. Because there are multiple services, the service name and host name must be specified explicitly. The Extractor connection of one of the fan-in hosts:

```
shell> trepctl -service alpha -host host1 status
Processing status command...
NAME                     VALUE
----                     -----
appliedLastEventId     : mysql-bin.000012:0000000000000418;0
appliedLastSeqno       : 0
appliedLatency         : 1.194
channels               : 1
clusterName            : alpha
currentEventId         : mysql-bin.000012:0000000000000418
currentTimeMillis      : 1375451438898
dataServerHost         : host1
extensions             :
latestEpochNumber      : 0
masterConnectUri       : thl://localhost:/
masterListenUri        : thl://host1:2112/
maximumStoredSeqNo     : 0
minimumStoredSeqNo     : 0
offlineRequests        : NONE
pendingError           : NONE
pendingErrorCode       : NONE
pendingErrorEventId    : NONE
pendingErrorSeqno      : -1
pendingExceptionMessage: NONE
pipelineSource         : jdbc:mysql:thin://host1:13306/
relativeLatency        : 6232.897
resourcePrecedence     : 99
rmiPort                : 10000
role                   : master
seqnoType              : java.lang.Long
serviceName            : alpha
serviceType            : local
simpleServiceName      : alpha
siteName               : default
sourceId               : host1
state                  : ONLINE
timeInStateSeconds     : 6231.881
transitioningTo        :
uptimeSeconds          : 6238.061
version                : Tungsten Replicator 7.1.2 build 81
Finished status command...
```

The corresponding Extractor service from the other host is `beta` on `host2`:

```
shell> trepctl -service beta -host host2 status
Processing status command...
NAME                     VALUE
----                     -----
appliedLastEventId     : mysql-bin.000012:0000000000000415;0
appliedLastSeqno       : 0
appliedLatency         : 0.941
channels               : 1
clusterName            : beta
currentEventId         : mysql-bin.000012:0000000000000415
currentTimeMillis      : 1375451493579
dataServerHost         : host2
extensions             :
latestEpochNumber      : 0
masterConnectUri       : thl://localhost:/
masterListenUri        : thl://host2:2112/
maximumStoredSeqNo     : 0
minimumStoredSeqNo     : 0
offlineRequests        : NONE
pendingError           : NONE
pendingErrorCode       : NONE
pendingErrorEventId    : NONE
pendingErrorSeqno      : -1
pendingExceptionMessage: NONE
pipelineSource         : jdbc:mysql:thin://host2:13306/
relativeLatency        : 6286.579
resourcePrecedence     : 99
rmiPort                : 10000
role                   : master
seqnoType              : java.lang.Long
serviceName            : beta
```

```
serviceType            : local
simpleServiceName      : beta
siteName               : default
sourceId               : host2
state                  : ONLINE
timeInStateSeconds     : 6285.823
transitioningTo        :
uptimeSeconds          : 6291.053
version                : Tungsten Replicator 7.1.2 build 81
Finished status command...
```

Note that because this is a fan-in topology, the sequence numbers and applied sequence numbers will be different for each service, as each service is independently storing data within the fan-in hub database.

The following sequence number combinations should match between the different hosts on each service:

| Extractor Service | Source Host | Target Host |
|---|---|---|
| `alpha` | `host1` | `host3` |
| `beta` | `host1` | `host3` |

The sequence numbers between `host1` and `host2` will not match, as they are two independent services.

For more information on using trepctl, see Section 8.20, "The trepctl Command".

Definitions of the individual field descriptions in the above example output can be found in Section E.2, "Generated Field Reference".

For more information on management and operational detailed for managing your cluster installation, see Chapter 7, *Operations Guide*.

# 5.3. Deploying Multiple Replicators on a Single Host

It is possible to install multiple replicators on the same host. This can be useful, either when building complex topologies with multiple services, and in hetereogenous environments where you are reading from one database and writing to another that may be installed on the same single server.

When installing multiple replicator services on the same host, different values must be set for the following configuration parameters:

## 5.3.1. Preparing Multiple Replicators

Before continuing with deployment you will need the following:

1. The name to use for the service.
2. The list of datasources in the service. These are the servers which will be running MySQL.
3. The username and password of the MySQL replication user.

All servers must be prepared with the proper prerequisites. See Appendix B, *Prerequisites* for additional details.

- RMI network port used for communicating with the replicator service.

  Set through the `--rmi-port` [422] parameter to tpm. Note that RMI ports are configured in pairs; the default port is 10000, port 10001 is used automatically. When specifying an alternative port, the subsequent port must also be available. For example, specifying port 10002 also requires 10003.

- THL network port used for exchanging THL data.

  Set through the `--thl-port` [429] parameter to tpm. The default THL port is 2112. This option is required for services operating as Extractors.

- Extractor THL port, i.e. the port from which an Applier will read THL events from the Extractor

  Set through the `--master-thl-port` [413] parameter to tpm. When operating as an Applier, the explicit THL port should be specified to ensure that you are connecting to the THL port correctly.

- Extractor hostname

  Set through the `--master-thl-host` [412] parameter to tpm. This is optional if the Extractor hostname has been configured correctly through the `--master` [412] parameter.

- Installation directory used when the replicator is installed.

  Set through the `--install-directory` [409] or `--install-directory` [409] parameters to tpm. This directory must have been created, and be configured with suitable permissions before installation starts. For more information, see Section B.3.3, "Directory Locations and Configuration".

## 5.3.2. Install Multiple Replicators

For example, to create two services, one that reads from MySQL and another that writes to MongoDB on the same host:

1. Install the Tungsten Replicator package or download the Tungsten Replicator tarball, and unpack it:

```
shell> cd /opt/continuent/software
shell> tar zxf tungsten-replicator-7.1.2-81.tar.gz
```

2. Create the proper directories with appropriate ownership and permissions:

```
shell> sudo mkdir /opt/applier /opt/extractor
shell> sudo chown tungsten: /opt/applier/ /opt/extractor/
shell> sudo chmod 700 /opt/applier/ /opt/extractor/
```

3. Change to the Tungsten Replicator directory:

```
shell> cd tungsten-replicator-7.1.2-81
```

4. Extractor reading from MySQL (Click link to switch examples between Staging Method or INI Method):

   Show Staging

   Show INI

```
shell> ./tools/tpm configure defaults \
    --reset \
    --install-directory=/opt/extractor \
    --user=tungsten \
    --profile-script=~/.bash_profile \
    --mysql-allow-intensive-checks=true \
    --disable-security-controls=true \
    --executable-prefix=ext \
    --rest-api-admin-user=apiuser \
    --rest-api-admin-pass=secret

shell> ./tools/tpm configure alpha \
    --master=offboardhost \
    --members=offboardhost \
    --enable-heterogeneous-service=true \
    --replication-port=3306 \
    --replication-user=tungsten_alpha \
    --replication-password=secret \
    --datasource-mysql-conf=/etc/my.cnf \
    --svc-extractor-filters=colnames,pkey \
    --property=replicator.filter.pkey.addColumnsToDeletes=true \
    --property=replicator.filter.pkey.addPkeyToInserts=true \
    --mysql-enable-enumtostring=true \
    --mysql-enable-settostring=true \
    --mysql-use-bytes-for-string=false
```

```
shell> vi /etc/tungsten/tungsten.ini
```

```
[defaults]
install-directory=/opt/extractor
user=tungsten
profile-script=~/.bash_profile
mysql-allow-intensive-checks=true
disable-security-controls=true
executable-prefix=ext
rest-api-admin-user=apiuser
rest-api-admin-pass=secret

[alpha]
master=offboardhost
members=offboardhost
enable-heterogeneous-service=true
replication-port=3306
replication-user=tungsten_alpha
replication-password=secret
datasource-mysql-conf=/etc/my.cnf
svc-extractor-filters=colnames,pkey
property=replicator.filter.pkey.addColumnsToDeletes=true
property=replicator.filter.pkey.addPkeyToInserts=true
mysql-enable-enumtostring=true
mysql-enable-settostring=true
mysql-use-bytes-for-string=false
```

   Configuration group `defaults`

   The description of each of the options is shown below; click the icon to hide this detail:

Click the icon to show a detailed description of each argument.

- `--reset` [422]

  `reset` [422]

  For staging configurations, deletes all pre-existing configuration information between updating with the new configuration values.

- `--install-directory=/opt/extractor` [409]

  `install-directory=/opt/extractor` [409]

  Path to the directory where the active deployment will be installed. The configured directory will contain the software, THL and relay log information unless configured otherwise.

- `--user=tungsten` [429]

  `user=tungsten` [429]

  System User

- `--profile-script=~/.bash_profile` [417]

  `profile-script=~/.bash_profile` [417]

  Append commands to include env.sh in this profile script

- `--mysql-allow-intensive-checks=true` [413]

  `mysql-allow-intensive-checks=true` [413]

  For MySQL installation, enables detailed checks on the supported data types within the MySQL database to confirm compatibility. This includes checking each table definition individually for any unsupported data types.

- `--disable-security-controls=true` [404]

  `disable-security-controls=true` [404]

  Disables all forms of security, including SSL, TLS and authentication

- `--executable-prefix=ext` [408]

  `executable-prefix=ext` [408]

  When enabled, the supplied prefix is added to each command alias that is generated for a given installation. This enables multiple installations to co-exist and and be accessible through a unique alias. For example, if the executable prefix is configured as `east`, then an alias for the installation to trepctl will be created as east_trepctl.

  Alias information for executable prefix data is stored within the `$CONTINUENT_ROOT/share/aliases.sh` file for each installation.

- `--rest-api-admin-user=apiuser` [422]

  `rest-api-admin-user=apiuser` [422]
  Optional: Must be specified along with rest-api-admin-pass if you wish to access the full API features and use the Dashboard GUI for cluster installations.

- `--rest-api-admin-pass=secret` [422]

  `rest-api-admin-pass=secret` [422]
  Optional: Must be specified along with rest-api-admin-user if you wish to access the full API features.

Configuration group **`alpha`**

The description of each of the options is shown below; click the icon to hide this detail:

Click the icon to show a detailed description of each argument.

- `--master=offboardhost` [412]

  `master=offboardhost` [412]

  The hostname of the primary (extractor) within the current service.

- `--members=offboardhost` [413]

  `members=offboardhost` [413]

  Hostnames for the dataservice members

- `--enable-heterogeneous-service=true` [406]

  `enable-heterogeneous-service=true` [406]

  - On a Primary
    - `--mysql-use-bytes-for-string` [414] is set to false.
    - `colnames` filter is enabled (in the `binlog-to-q` stage to add column names to the THL information.
    - `pkey` filter is enabled (in the `binlog-to-q` and `q-to-dbms` stage), with the *`addPkeyToInserts`* and *`addColumnsToDeletes`* filter options set to false.
    - `enumtostring` filter is enabled (in the `q-to-thl` stage), to translate `ENUM` values to their string equivalents.
    - `settostring` filter is enabled (in the `q-to-thl` stage), to translate `SET` values to their string equivalents.
  - On a Replica
    - `--mysql-use-bytes-for-string` [414] is set to true.
    - `pkey` filter is enabled (`q-to-dbms` stage).

- `--replication-port=3306` [421]

  `replication-port=3306` [421]

  The network port used to connect to the database server. The default port used depends on the database being configured.

- `--replication-user=tungsten_alpha` [421]

  `replication-user=tungsten_alpha` [421]

  For databases that required authentication, the username to use when connecting to the database using the corresponding connection method (native, JDBC, etc.).

- `--replication-password=secret` [420]

  `replication-password=secret` [420]

  The password to be used when connecting to the database using the corresponding `--replication-user` [421].

- `--datasource-mysql-conf=/etc/my.cnf` [401]

  `datasource-mysql-conf=/etc/my.cnf` [401]

  MySQL config file

- `--svc-extractor-filters=colnames,pkey` [425]

  `svc-extractor-filters=colnames,pkey` [425]

  Replication service extractor filters

- `--mysql-enable-enumtostring=true` [414]

  `mysql-enable-enumtostring=true` [414]

  Enable a filter to convert ENUM values to strings

- `--mysql-enable-settostring=true` [414]

  `mysql-enable-settostring=true` [414]

  Enable a filter to convert SET types to strings



- `--mysql-use-bytes-for-string=false` [414]

`mysql-use-bytes-for-string=false` [414]

Transfer strings as their byte representation?

This is a standard configuration using the default ports, with the directory `/opt/extractor`.

5. Applier for writing to MongoDB (Click link to switch examples between Staging Method or INI Method):

   Show Staging

   Show INI

```
shell> ./tools/tpm configure defaults \
    --reset \
    --install-directory=/opt/applier \
    --profile-script=~/.bash_profile \
    --skip-validation-check=InstallerMasterSlaveCheck \
    --executable-prefix=app \
    --rest-api-admin-user=apiuser \
    --rest-api-admin-pass=secret

shell> ./tools/tpm configure alpha \
    --master=localhost \
    --members=localhost \
    --role=slave \
    --datasource-type=mongodb \
    --replication-user=tungsten \
    --replication-password=secret \
    --rmi-port=10002 \
    --master-thl-port=2112 \
    --master-thl-host=localhost \
    --thl-port=2113
```

```
shell> vi /etc/tungsten/tungsten.ini
```

```
[defaults]
install-directory=/opt/applier
profile-script=~/.bash_profile
skip-validation-check=InstallerMasterSlaveCheck
executable-prefix=app
rest-api-admin-user=apiuser
rest-api-admin-pass=secret

[alpha]
master=localhost
members=localhost
role=slave
datasource-type=mongodb
replication-user=tungsten
replication-password=secret
rmi-port=10002
master-thl-port=2112
master-thl-host=localhost
thl-port=2113
```

   Configuration group **`defaults`**

   The description of each of the options is shown below; click the icon to hide this detail:

   Click the icon to show a detailed description of each argument.

   - `--reset` [422]

     `reset` [422]

     For staging configurations, deletes all pre-existing configuration information between updating with the new configuration values.

   - `--install-directory=/opt/applier` [409]

     `install-directory=/opt/applier` [409]

     Path to the directory where the active deployment will be installed. The configured directory will contain the software, THL and relay log information unless configured otherwise.

   - `--profile-script=~/.bash_profile` [417]

     `profile-script=~/.bash_profile` [417]

Append commands to include env.sh in this profile script

- `--skip-validation-check=InstallerMasterSlaveCheck` [369]

  `skip-validation-check=InstallerMasterSlaveCheck` [369]

  The `--skip-validation-check` [369] disables a given validation check. If any validation check fails, the installation, validation or configuration will automatically stop.

  > **Warning**
  >
  > Using this option enables you to bypass the specified check, although skipping a check may lead to an invalid or non-working configuration.

  You can identify a given check if an error or warning has been raised during configuration. For example, the default table type check:

  ```
  ...
  ERROR >> centos >> The datasource root@centos:3306 (WITH PASSWORD) »
   uses MyISAM as the default storage engine (MySQLDefaultTableTypeCheck)
  ...
  ```

  The check in this case is `MySQLDefaultTableTypeCheck` [380], and could be ignored using `--skip-validation-check=MySQLDefaultTableTypeCheck` [369].

  Setting both `--skip-validation-check` [369] and `--enable-validation-check` [367] is equivalent to explicitly disabling the specified check.

- `--executable-prefix=app` [408]

  `executable-prefix=app` [408]

  When enabled, the supplied prefix is added to each command alias that is generated for a given installation. This enables multiple installations to co-exist and and be accessible through a unique alias. For example, if the executable prefix is configured as `east`, then an alias for the installation to trepctl will be created as east_trepctl.

  Alias information for executable prefix data is stored within the `$CONTINUENT_ROOT/share/aliases.sh` file for each installation.

- `--rest-api-admin-user=apiuser` [422]

  `rest-api-admin-user=apiuser` [422]
  Optional: Must be specified along with rest-api-admin-pass if you wish to access the full API features and use the Dashboard GUI for cluster installations.

- `--rest-api-admin-pass=secret` [422]

  `rest-api-admin-pass=secret` [422]
  Optional: Must be specified along with rest-api-admin-user if you wish to access the full API features.

Configuration group **`alpha`**

The description of each of the options is shown below; click the icon to hide this detail:

Click the icon to show a detailed description of each argument.

- `--master=localhost` [412]

  `master=localhost` [412]

  The hostname of the primary (extractor) within the current service.

- `--members=localhost` [413]

  `members=localhost` [413]

  Hostnames for the dataservice members

- `--role=slave` [422]

  `role=slave` [422]

  What is the replication role for this service?

- `--datasource-type=mongodb` [402]

datasource-type=mongodb [402]

Database type

- --replication-user=tungsten [421]

  replication-user=tungsten [421]

  For databases that required authentication, the username to use when connecting to the database using the corresponding connection method (native, JDBC, etc.).

- --replication-password=secret [420]

  replication-password=secret [420]

  The password to be used when connecting to the database using the corresponding --replication-user [421].

- --rmi-port=10002 [422]

  rmi-port=10002 [422]

  Replication RMI listen port

- --master-thl-port=2112 [413]

  master-thl-port=2112 [413]

  Primary THL Port

- --master-thl-host=localhost [412]

  master-thl-host=localhost [412]

  Primary THL Hostname

- --thl-port=2113 [429]

  thl-port=2113 [429]

  Port to use for THL Operations

In this configuration, the Extractor THL port is specified explicitly, along with the THL port used by this replicator, the RMI port used for administration, and the installation directory /opt/applier.

6. Run tpm to install the software

```
shell > ./tools/tpm install
```

During the startup and installation, tpm will notify you of any problems that need to be fixed before the service can be correctly installed and started. If start-and-report [424] is set and the service starts correctly, you should see the configuration and current status of the service.

7. Initialize your PATH and environment.

```
shell > source /opt/extractor/share/env.sh
shell > source /opt/applier/share/env.sh
```

8. Check the replication status.

When multiple replicators have been installed, checking the replicator status through trepctl depends on the replicator executable location used. If /opt/extractor/tungsten/tungsten-replicator/bin/trepctl, the extractor service status will be reported. If /opt/applier/tungsten/tungsten-replicator/bin/trepctl is used, then the applier service status will be reported.

To make things easier, in the config examples above executable-prefix [408] has been used, which will set up OS aliases. These aliases are setup when you source the relevant env.sh files, this will also happen by default when you login to the host providing profile-script [417] has been specified

The use of the prefix and aliases, then simplifies the use of all executables, for example, based on the setting of executable-prefix [408] in the above config examples, to report the status of the extractor, you can execute:

```
shell> ext_trepctl status
```

Or to check the applier service:

```
shell> app_trepctl status
```

Alternatively, a specific replicator can be checked by explicitly specifying the RMI port of the service. For example, to check the extractor service:

```
shell> trepctl -port 10000 status
```

Or to check the applier service:

```
shell> trepctl -port 10002 status
```

When an explicit port has been specified in this way, the executable used is irrelevant. Any valid trepctl instance will work.

Further, either path may be used to get a summary view using multi_trepctl:

```
shell> /opt/extractor/tungsten/tungsten-replicator/scripts/multi_trepctl
| host   | servicename | role   | state   | appliedlastseqno | appliedlatency |
| host1  | extractor   | master | ONLINE  |                0 |          1.724 |
| host1  | applier     | slave  | ONLINE  |                0 |          0.000 |
```

### 5.3.3. Best Practices: Multiple Replicators

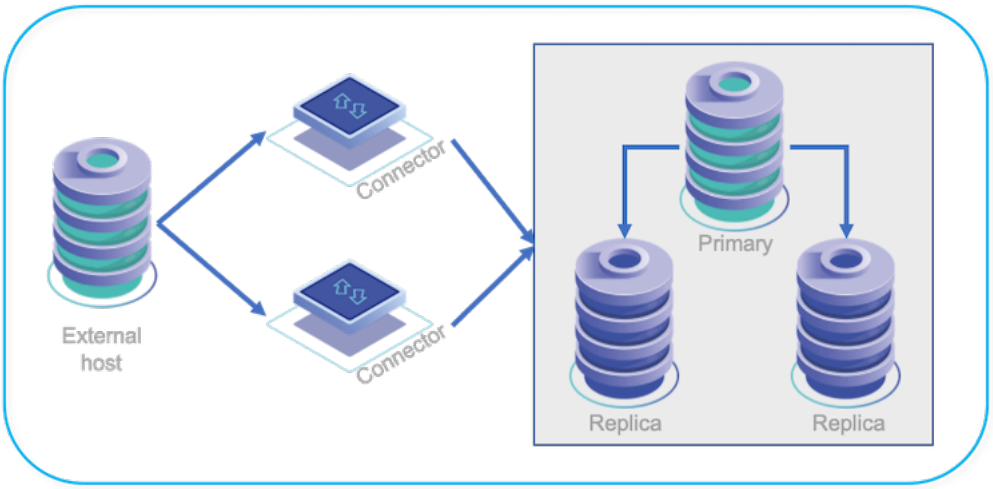
Follow the guidelines in Section 2.2, “Best Practices”.

## 5.4. Replicating Data Into an Existing Dataservice

If you have an existing dataservice, data can be replicated from a standalone MySQL server into the service. The replication is configured by creating a service that reads from the standalone MySQL server and writes into the Primary of the target dataservice. By writing this way, changes are replicated to the Primary and Replica in the new deployment.

Additionally, using a replicator that writes data into an existing data service can be used when migrating from an existing service into a new Tungsten Cluster service.

Figure 5.2. Topologies: Replicating into a Dataservice




In order to configure this deployment, there are two steps:

1. Create a new replicator that reads this data and writes the replicated data into the Primary of the destination dataservice.
2. Create a new replicator that reads the binary logs directly from the external MySQL service through the Primary of the destination dataservice

There are also the following requirements:

- The host on which you want to replicate to must have Tungsten Replicator 5.3.0 or later.
- Hosts on both the replicator and cluster must be able to communicate with each other.
- The replication user on the source host must have the `RELOAD`, `REPLICATION SLAVE`, and `REPLICATION CLIENT GRANT` privileges.
- Replicator must be able to connect as the `tungsten` user to the databases within the cluster.

Install the Tungsten Replicator package (see Section 2.1.2, “Using the RPM package files”), or download the compressed tarball and unpack it on `host1`:

```
shell> cd /opt/replicator/software
shell> tar zxf tungsten-replicator-7.1.2-81.tar.gz
```

Change to the Tungsten Replicator staging directory:

```
shell> cd tungsten-replicator-7.1.2-81
```

Configure the replicator on `host1`

First we configure the defaults and a cluster alias that points to the Primaries and Replicas within the current Tungsten Cluster service that you are replicating from:

Click the link below to switch examples between Staging and INI methods

Show Staging

Show INI

```
shell> ./tools/tpm configure defaults \
    --install-directory=/opt/replicator \
    --rmi-port=10002 \
    --user=tungsten \
    --replication-user=tungsten \
    --replication-password=secret \
    --skip-validation-check=MySQLNoMySQLReplicationCheck \
    --rest-api-admin-user=apiuser \
    --rest-api-admin-pass=secret

shell> ./tools/tpm configure beta \
    --topology=direct \
    --master=host1 \
    --direct-datasource-host=host3 \
    --thl-port=2113
```

```
shell> vi /etc/tungsten/tungsten.ini
```

```
[defaults]
install-directory=/opt/replicator
rmi-port=10002
user=tungsten
replication-user=tungsten
replication-password=secret
skip-validation-check=MySQLNoMySQLReplicationCheck
rest-api-admin-user=apiuser
rest-api-admin-pass=secret

[beta]
topology=direct
master=host1
direct-datasource-host=host3
thl-port=2113
```

Configuration group **`defaults`**

The description of each of the options is shown below; click the icon to hide this detail:

Click the icon to show a detailed description of each argument.

- `--install-directory=/opt/replicator` [409]

  `install-directory=/opt/replicator` [409]

  Path to the directory where the active deployment will be installed. The configured directory will contain the software, THL and relay log information unless configured otherwise.

- `--rmi-port=10002` [422]

  `rmi-port=10002` [422]

  Replication RMI listen port

- `--user=tungsten` [429]

  `user=tungsten` [429]

  System User

- `--replication-user=tungsten` [421]

  `replication-user=tungsten` [421]

  For databases that required authentication, the username to use when connecting to the database using the corresponding connection method (native, JDBC, etc.).

- `--replication-password=secret` [420]

  `replication-password=secret` [420]

  The password to be used when connecting to the database using the corresponding `--replication-user` [421].

- `--skip-validation-check=MySQLNoMySQLReplicationCheck` [369]

  `skip-validation-check=MySQLNoMySQLReplicationCheck` [369]

  The `--skip-validation-check` [369] disables a given validation check. If any validation check fails, the installation, validation or configuration will automatically stop.

  > **Warning**
  >
  > Using this option enables you to bypass the specified check, although skipping a check may lead to an invalid or non-working configuration.

  You can identify a given check if an error or warning has been raised during configuration. For example, the default table type check:

  ```
  ...
  ERROR >> centos >> The datasource root@centos:3306 (WITH PASSWORD) »
   uses MyISAM as the default storage engine (MySQLDefaultTableTypeCheck)
  ...
  ```

  The check in this case is `MySQLDefaultTableTypeCheck` [380], and could be ignored using `--skip-validation-check=MySQLDefaultTableType-Check` [369].

  Setting both `--skip-validation-check` [369] and `--enable-validation-check` [367] is equivalent to explicitly disabling the specified check.

- `--rest-api-admin-user=apiuser` [422]

  `rest-api-admin-user=apiuser` [422]
  Optional: Must be specified along with rest-api-admin-pass if you wish to access the full API features and use the Dashboard GUI for cluster installations.

- `--rest-api-admin-pass=secret` [422]

  `rest-api-admin-pass=secret` [422]
  Optional: Must be specified along with rest-api-admin-user if you wish to access the full API features.

Configuration group **`beta`**

The description of each of the options is shown below; click the icon to hide this detail:

Click the icon to show a detailed description of each argument.

- `--topology=direct` [429]

  `topology=direct` [429]

  Replication topology for the dataservice.

- `--master=host1` [412]

  `master=host1` [412]

  The hostname of the primary (extractor) within the current service.

- `--direct-datasource-host=host3` [403]

  `direct-datasource-host=host3` [403]

  Database server hostname

- `--thl-port=2113` [429]

`thl-port=2113` [429]

Port to use for THL Operations

This creates a configuration that specifies that the topology should read directly from the source host, `host3`, writing directly to `host1`. An alternative THL port is provided to ensure that the THL listener is not operating on the same network port as the original.

Now install the service, which will create the replicator reading direct from `host3` into `host1`:

```
shell> ./tools/tpm install
```

If the installation process fails, check the output of the `/tmp/tungsten-configure.log` file for more information about the root cause.

Once the installation has been completed, you must update the position of the replicator so that it points to the correct position within the source database to prevent errors during replication. If the replication is being created as part of a migration process, determine the position of the binary log from the external replicator service used when the backup was taken. For example:

```
mysql> show master status;
*************************** 1. row ***************************
            File: mysql-bin.000026
        Position: 1311
    Binlog_Do_DB:
Binlog_Ignore_DB:
1 row in set (0.00 sec)
```

Use dsctl set to update the replicator position to point to the Primary log position:

```
shell> /opt/replicator/tungsten/tungsten-replicator/bin/dsctl -service beta set \
    -reset -seqno 0 -epoch 0 \
    -source-id host3 -event-id mysql-bin.000026:1311
```

Now start the replicator:

```
shell> /opt/replicator/tungsten/tungsten-replicator/bin/replicator start
```

Replication status should be checked by explicitly using the servicename and/or RMI port:

```
shell> /opt/replicator/tungsten/tungsten-replicator/bin/trepctl -service beta status
Processing status command...
NAME                     VALUE
----                     -----
appliedLastEventId     : mysql-bin.000026:0000000000001311;1252
appliedLastSeqno       : 5
appliedLatency         : 0.748
channels               : 1
clusterName            : beta
currentEventId         : mysql-bin.000026:0000000000001311
currentTimeMillis      : 1390410611881
dataServerHost         : host1
extensions             :
host                   : host3
latestEpochNumber      : 1
masterConnectUri       : thl://host3:2112/
masterListenUri        : thl://host1:2113/
maximumStoredSeqNo     : 5
minimumStoredSeqNo     : 0
offlineRequests        : NONE
pendingError           : NONE
pendingErrorCode       : NONE
pendingErrorEventId    : NONE
pendingErrorSeqno      : -1
pendingExceptionMessage: NONE
pipelineSource         : jdbc:mysql:thin://host3:13306/
relativeLatency        : 8408.881
resourcePrecedence     : 99
rmiPort                : 10000
role                   : master
seqnoType              : java.lang.Long
serviceName            : beta
serviceType            : local
simpleServiceName      : beta
siteName               : default
sourceId               : host3
state                  : ONLINE
timeInStateSeconds     : 8408.21
transitioningTo        :
uptimeSeconds          : 8409.88
useSSLConnection       : false
version                : Tungsten Replicator 7.1.2 build 81
Finished status command...
```

# 5.5. Deploying Parallel Replication

Parallel apply is an important technique for achieving high speed replication and curing Replica lag. It works by spreading updates to Replicas over multiple threads that split transactions on each schema into separate processing streams. This in turn spreads I/O activity across many threads, which results in faster overall updates on the Replica. In ideal cases throughput on Replicas may improve by up to 5 times over single-threaded MySQL native replication.

> **Note**
>
> It is worth noting that the only thing Tungsten parallelizes is applying transactions to Replicas. All other operations in each replication service are single-threaded.

## 5.5.1. Application Prerequisites for Parallel Replication

Parallel replication works best on workloads that meet the following criteria:

- ROW based binary logging must be enabled in the MySQL database.
- Data are stored in independent schemas. If you have 100 customers per server with a separate schema for each customer, your application is a good candidate.
- Transactions do not span schemas. Tungsten serializes such transactions, which is to say it stops parallel apply and runs them by themselves. If more than 2-3% of transactions are serialized in this way, most of the benefits of parallelization are lost.
- Workload is well-balanced across schemas.
- The Replica host(s) are capable and have free memory in the OS page cache.
- The host on which the Replica runs has a sufficient number of cores to operate a large number of Java threads.
- Not all workloads meet these requirements. If your transactions are within a single schema only, you may need to consider different approaches, such as Replica prefetch. Contact Continuent for other suggestions.

Parallel replication does not work well on underpowered hosts, such as Amazon m1.small instances. In fact, any host that is already I/O bound under single-threaded replication will typical will not show much improvement with parallel apply.

## 5.5.2. Enabling Parallel Apply During Install

Parallel apply is enabled using the `svc-parallelization-type` [425] and `channels` [398] options of tpm. The parallelization type defaults to *`none`* which is to say that parallel apply is disabled. You should set it to *`disk`* [168]. The `channels` [398] option sets the the number of channels (i.e., threads) you propose to use for applying data. Here is a code example of a MySQL Applier installation with parallel apply enabled. The Replica will apply transactions using 30 channels.

Show Staging

Show INI

```
shell> ./tools/tpm configure defaults \
    --reset \
    --install-directory=/opt/continuent \
    --user=tungsten \
    --mysql-allow-intensive-checks=true \
    --profile-script=~/.bash_profile \
    --start-and-report=true

shell> ./tools/tpm configure alpha \
    --master=sourcehost \
    --members=localhost,sourcehost \
    --datasource-type=mysql \
    --replication-user=tungsten \
    --replication-password=secret \
    --svc-parallelization-type=disk \
    --channels=10
```

```
shell> vi /etc/tungsten/tungsten.ini
```

```
[defaults]
install-directory=/opt/continuent
user=tungsten
mysql-allow-intensive-checks=true
profile-script=~/.bash_profile
start-and-report=true
```

```
[alpha]
master=sourcehost
members=localhost,sourcehost
datasource-type=mysql
replication-user=tungsten
replication-password=secret
svc-parallelization-type=disk
channels=10
```

Configuration group **defaults**

The description of each of the options is shown below; click the icon to hide this detail:

Click the icon to show a detailed description of each argument.

- `--reset` [422]

  `reset` [422]

  For staging configurations, deletes all pre-existing configuration information between updating with the new configuration values.

- `--install-directory=/opt/continuent` [409]

  `install-directory=/opt/continuent` [409]

  Path to the directory where the active deployment will be installed. The configured directory will contain the software, THL and relay log information unless configured otherwise.

- `--user=tungsten` [429]

  `user=tungsten` [429]

  System User

- `--mysql-allow-intensive-checks=true` [413]

  `mysql-allow-intensive-checks=true` [413]

  For MySQL installation, enables detailed checks on the supported data types within the MySQL database to confirm compatibility. This includes checking each table definition individually for any unsupported data types.

- `--profile-script=~/.bash_profile` [417]

  `profile-script=~/.bash_profile` [417]

  Append commands to include env.sh in this profile script

- `--start-and-report=true` [424]

  `start-and-report=true` [424]

  Start the services and report out the status after configuration

Configuration group **alpha**

The description of each of the options is shown below; click the icon to hide this detail:

Click the icon to show a detailed description of each argument.

- `--master=sourcehost` [412]

  `master=sourcehost` [412]

  The hostname of the primary (extractor) within the current service.

- `--members=localhost,sourcehost` [413]

  `members=localhost,sourcehost` [413]

  Hostnames for the dataservice members

- `--datasource-type=mysql` [402]

  `datasource-type=mysql` [402]

Database type

- `--replication-user=tungsten` [421]

  `replication-user=tungsten` [421]

  For databases that required authentication, the username to use when connecting to the database using the corresponding connection method (native, JDBC, etc.).

- `--replication-password=secret` [420]

  `replication-password=secret` [420]

  The password to be used when connecting to the database using the corresponding `--replication-user` [421].

- `--svc-parallelization-type=disk` [425]

  `svc-parallelization-type=disk` [425]

  Method for implementing parallel apply

- `--channels=10` [398]

  `channels=10` [398]

  Number of replication channels to use for parallel apply.

If the installation process fails, check the output of the `/tmp/tungsten-configure.log` file for more information about the root cause.

There are several additional options that default to reasonable values. You may wish to change them in special cases.

- `buffer-size` — Sets the replicator block commit size, which is the number of transactions to commit at once on Replicas. Values up to 100 are normally fine.
- `native-slave-takeover` [415] — Used to allow Tungsten to take over from native MySQL replication and parallelize it. See here for more.

You can check the number of active channels on a Replica by looking at the "channels" property once the replicator restarts.

```
Replica shell> trepctl -service alpha status| grep channels
channels               : 10
```

### Important

The channel count for a Primary will ALWAYS be 1 because extraction is single-threaded:

```
Primary shell> trepctl -service alpha status| grep channels
channels               : 1
```

### Warning

Enabling parallel apply will dramatically increase the number of connections to the database server.

Typically the calculation on a Replica would be: Connections = Channel_Count x Sevice_Count x 2, so for a 4-way Composite Composite Active/Active topology with 30 channels there would be 30 x 4 x 2 = 240 connections required for the replicator alone, not counting application traffic.

You may display the currently used number of connections in MySQL:

```
mysql> SHOW STATUS LIKE 'max_used_connections';
+----------------------+-------+
| Variable_name        | Value |
+----------------------+-------+
| Max_used_connections | 190   |
+----------------------+-------+
1 row in set (0.00 sec)
```

Below are suggestions for how to change the maximum connections setting in MySQL both for the running instance as well as at startup:

```
mysql> SET GLOBAL max_connections = 512;

mysql> SHOW VARIABLES LIKE 'max_connections';
+-----------------+-------+
| Variable_name   | Value |
```

```
+-----------------+-------+
| max_connections | 512   |
+-----------------+-------+
1 row in set (0.00 sec)

shell> vi /etc/my.cnf
#max_connections = 151
max_connections = 512
```

## 5.5.3. Channels

Channels and Parallel Apply

Parallel apply works by using multiple threads for the final stage of the replication pipeline. These threads are known as channels. Restart points for each channel are stored as individual rows in table `trep_commit_seqno` if you are applying to a relational DBMS server, including MySQL, Oracle, and data warehouse products like Vertica.

When you set the `channels` [398] argument, the tpm program configures the replication service to enable the requested number of channels. A value of 1 results in single-threaded operation.

Do not change the number of channels without setting the replicator offline cleanly. See the procedure later in this page for more information.

How Many Channels Are Enough?

Pick the smallest number of channels that loads the Replica fully. For evenly distributed workloads this means that you should increase channels so that more threads are simultaneously applying updates and soaking up I/O capacity. As long as each shard receives roughly the same number of updates, this is a good approach.

For unevenly distributed workloads, you may want to decrease channels to spread the workload more evenly across them. This ensures that each channel has productive work and minimizes the overhead of updating the channel position in the DBMS.

Once you have maximized I/O on the DBMS server leave the number of channels alone. Note that adding more channels than you have shards does not help performance as it will lead to idle channels that must update their positions in the DBMS even though they are not doing useful work. This actually slows down performance a little bit.

Effect of Channels on Backups

If you back up a Replica that operates with more than one channel, say 30, you can only restore that backup on another Replica that operates with the same number of channels. Otherwise, reloading the backup is the same as changing the number of channels without a clean offline.

When operating Tungsten Replicator in a Tungsten cluster, you should always set the number of channels to be the same for all replicators. Otherwise you may run into problems if you try to restore backups across MySQL instances that load with different locations.

If the replicator has only a single channel enabled, you can restore the backup anywhere. The same applies if you run the backup after the replicator has been taken offline cleanly.

## 5.5.4. Parallel Replication and Offline Operation

### 5.5.4.1. Clean Offline Operation

When you issue a trepctl offline command, Tungsten Replicator will bring all channels to the same point in the log and then go offline. This is known as going offline cleanly. When a Replica has been taken offline cleanly the following are true:

- The `trep_commit_seqno` table contains a single row
- The `trep_shard_channel` table is empty

When parallel replication is not enabled, you can take the replicator offline by stopping the replicator process. There is no need to issue a trepctl offline command first.

### 5.5.4.2. Tuning the Time to Go Offline Cleanly

Putting a replicator offline may take a while if the slowest and fastest channels are far apart, i.e., if one channel gets far ahead of another. The separation between channels is controlled by the *`maxOfflineInterval`* parameter, which defaults to 5 seconds. This sets the allowable distance between commit timestamps processed on different channels. You can adjust this value at installation or later. The following example shows how to change it after installation. This can be done at any time and does not require the replicator to go offline cleanly.

Click the link below to switch examples between Staging and INI methods...

Show Staging

Show INI

```
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-replicator-7.1.2-81

shell> echo The staging USER is `tpm query staging| cut -d: -f1 | cut -d@ -f1`
The staging USER is tungsten

shell> echo The staging HOST is `tpm query staging| cut -d: -f1 | cut -d@ -f2`
The staging HOST is db1

shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-replicator-7.1.2-81

shell> ssh {STAGING_USER}@{STAGING_HOST}
shell> cd {STAGING_DIRECTORY}
```

```
shell> ./tools/tpm configure alpha \
    --property=replicator.store.parallel-queue.maxOfflineInterval=30
```

Run the tpm command to update the software with the Staging-based configuration:

```
shell> ./tools/tpm update
```

For information about making updates when using a Staging-method deployment, please see Section 9.3.7, “Configuration Changes from a Staging Directory”.

```
shell> vi /etc/tungsten/tungsten.ini
```

```
[alpha]
...
property=replicator.store.parallel-queue.maxOfflineInterval=30
```

Run the tpm command to update the software with the INI-based configuration:

```
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-replicator-7.1.2-81

shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-replicator-7.1.2-81

shell> cd {STAGING_DIRECTORY}

shell> ./tools/tpm update
```

For information about making updates when using an INI file, please see Section 9.4.4, “Configuration Changes with an INI file”.

The offline interval is only the the approximate time that Tungsten Replicator will take to go offline. Up to a point, larger values (say 60 or 120 seconds) allow the replicator to parallelize in spite of a few operations that are relatively slow. However, the down side is that going offline cleanly can become quite slow.

### 5.5.4.3. Unclean Offline

If you need to take a replicator offline quickly, you can either stop the replicator process or issue the following command:

```
shell> trepctl offline -immediate
```

Both of these result in an unclean shutdown. However, parallel replication is completely crash-safe provided you use transactional table types like InnoDB, so you will be able to restart without causing Replica consistency problems.

**Warning**

> You must take the replicator offline cleanly to change the number of channels or when reverting to MySQL native replication. Failing to do so can result in errors when you restart replication.

## 5.5.5. Adjusting Parallel Replication After Installation

### 5.5.5.1. How to Enable Parallel Apply After Installation

To enable parallel replication after installation, take the replicator offline cleanly using the following command:

```
shell> trepctl offline
```

Modify the configuration to add two parameters:

Show Staging

Show INI

```
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-replicator-7.1.2-81

shell> echo The staging USER is `tpm query staging| cut -d: -f1 | cut -d@ -f1`
The staging USER is tungsten

shell> echo The staging HOST is `tpm query staging| cut -d: -f1 | cut -d@ -f2`
The staging HOST is db1

shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-replicator-7.1.2-81

shell> ssh {STAGING_USER}@{STAGING_HOST}
shell> cd {STAGING_DIRECTORY}
```

```
shell> ./tools/tpm configure defaults \
    --svc-parallelization-type=disk \
    --channels=10
```

Run the tpm command to update the software with the Staging-based configuration:

```
shell> ./tools/tpm update
```

For information about making updates when using a Staging-method deployment, please see Section 9.3.7, "Configuration Changes from a Staging Directory".

```
[defaults]
...
svc-parallelization-type=disk
channels=10
```

Run the tpm command to update the software with the INI-based configuration:

```
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-replicator-7.1.2-81

shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-replicator-7.1.2-81

shell> cd {STAGING_DIRECTORY}

shell> ./tools/tpm update
```

For information about making updates when using an INI file, please see Section 9.4.4, "Configuration Changes with an INI file".

### Note

You make use an actual data service name in place of the keyword `defaults`.

Signal the changes by a complete restart of the Replicator process:

```
shell> replicator restart
```

You can check the number of active channels on a Replica by looking at the "channels" property once the replicator restarts.

```
Replica shell> trepctl -service alpha status| grep channels
channels               : 10
```

### Important

The channel count for a Primary will ALWAYS be 1 because extraction is single-threaded:

```
Primary shell> trepctl -service alpha status| grep channels
channels               : 1
```

### Warning

Enabling parallel apply will dramatically increase the number of connections to the database server.

Typically the calculation on a Replica would be: Connections = Channel_Count x Sevice_Count x 2, so for a 4-way Composite Composite Active/Active topology with 30 channels there would be 30 x 4 x 2 = 240 connections required for the replicator alone, not counting application traffic.

You may display the currently used number of connections in MySQL:

```
mysql> SHOW STATUS LIKE 'max_used_connections';
+----------------------+-------+
| Variable_name        | Value |
+----------------------+-------+
| Max_used_connections | 190   |
+----------------------+-------+
1 row in set (0.00 sec)
```

Below are suggestions for how to change the maximum connections setting in MySQL both for the running instance as well as at startup:

```
mysql> SET GLOBAL max_connections = 512;

mysql> SHOW VARIABLES LIKE 'max_connections';
+-----------------+-------+
| Variable_name   | Value |
+-----------------+-------+
| max_connections | 512   |
+-----------------+-------+
1 row in set (0.00 sec)

shell> vi /etc/my.cnf
#max_connections = 151
max_connections = 512
```

## 5.5.5.2. How to Change Channels Safely

To change the number of channels you must take the replicator offline cleanly using the following command:

```
shell> trepctl offline
```

This command brings all channels up the same transaction in the log, then goes offline. If you look in the `trep_commit_seqno` table, you will notice only a single row, which shows that updates to the Replica have been completely serialized to a single point. At this point you may safely reconfigure the number of channels on the replicator, for example using the following command:

Click the link below to switch examples between Staging and INI methods...

Show Staging

Show INI

```
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-replicator-7.1.2-81

shell> echo The staging USER is `tpm query staging| cut -d: -f1 | cut -d@ -f1`
The staging USER is tungsten

shell> echo The staging HOST is `tpm query staging| cut -d: -f1 | cut -d@ -f2`
The staging HOST is db1

shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-replicator-7.1.2-81

shell> ssh {STAGING_USER}@{STAGING_HOST}
shell> cd {STAGING_DIRECTORY}
```

```
shell> ./tools/tpm configure alpha \
    --channels=5
```

Run the tpm command to update the software with the Staging-based configuration:

```
shell> ./tools/tpm update
```

For information about making updates when using a Staging-method deployment, please see Section 9.3.7, “Configuration Changes from a Staging Directory”.

```
[alpha]
...
channels=5
```

Run the tpm command to update the software with the INI-based configuration:

```
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-replicator-7.1.2-81

shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-replicator-7.1.2-81

shell> cd {STAGING_DIRECTORY}
```

```
shell> ./tools/tpm update
```

For information about making updates when using an INI file, please see Section 9.4.4, “Configuration Changes with an INI file”.

You can check the number of active channels on a Replica by looking at the "channels" property once the replicator restarts.

If you attempt to reconfigure channels without going offline cleanly, Tungsten Replicator will signal an error when you attempt to go online with the new channel configuration. The cure is to revert to the previous number of channels, go online, and then go offline cleanly. Note that attempting to clean up the `trep_commit_seqno` and `trep_shard_channel` tables manually can result in your Replicas becoming inconsistent and requiring full resynchronization. You should only do such cleanup under direction from Continuent support.

#### Warning

> Failing to follow the channel reconfiguration procedure carefully may result in your Replicas becoming inconsistent or failing. The cure is usually full resynchronization, so it is best to avoid this if possible.

### 5.5.5.3. How to Disable Parallel Replication Safely

The following steps describe how to gracefully disable parallel apply replication.

#### Replication Graceful Offline (critical first step)

To disable parallel apply, you must first take the replicator offline cleanly using the following command:

```
shell> trepctl offline
```

This command brings all channels up the same transaction in the log, then goes offline. If you look in the `trep_commit_seqno` table, you will notice only a single row, which shows that updates to the Replica have been completely serialized to a single point. At this point you may safely disable parallel apply on the replicator, for example using the following command:

Click the link below to switch examples between Staging and INI methods...

Show Staging

Show INI

```
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-replicator-7.1.2-81

shell> echo The staging USER is `tpm query staging| cut -d: -f1 | cut -d@ -f1`
The staging USER is tungsten

shell> echo The staging HOST is `tpm query staging| cut -d: -f1 | cut -d@ -f2`
The staging HOST is db1

shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-replicator-7.1.2-81

shell> ssh {STAGING_USER}@{STAGING_HOST}
shell> cd {STAGING_DIRECTORY}
```

```
shell> ./tools/tpm configure alpha \
    --svc-parallelization-type=none \
    --channels=1
```

Run the tpm command to update the software with the Staging-based configuration:

```
shell> ./tools/tpm update
```

For information about making updates when using a Staging-method deployment, please see Section 9.3.7, “Configuration Changes from a Staging Directory”.

```
[alpha]
...
svc-parallelization-type=none
channels=1
```

Run the tpm command to update the software with the INI-based configuration:

```
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-replicator-7.1.2-81

shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-replicator-7.1.2-81

shell> cd {STAGING_DIRECTORY}
```

```
shell> ./tools/tpm update
```

For information about making updates when using an INI file, please see Section 9.4.4, “Configuration Changes with an INI file”.

## Verification

You can check the number of active channels on a Replica by looking at the "channels" property once the replicator restarts.

```
shell> trepctl -service alpha status| grep channels
channels               : 1
```

## Notes and Warnings

If you attempt to reconfigure channels without going offline cleanly, Tungsten Replicator will signal an error when you attempt to go online with the new channel configuration. The cure is to revert to the previous number of channels, go online, and then go offline cleanly. Note that attempting to clean up the `trep_commit_seqno` and `trep_shard_channel` tables manually can result in your Replicas becoming inconsistent and requiring full resynchronization. You should only do such cleanup under direction from Continuent support.

### Warning

> Failing to follow the channel reconfiguration procedure carefully may result in your Replicas becoming inconsistent or failing. The cure is usually full resynchronization, so it is best to avoid this if possible.

# 5.5.5.4. How to Switch Parallel Queue Types Safely

As with channels you should only change the parallel queue type after the replicator has gone offline cleanly. The following example shows how to update the parallel queue type after installation:

Click the link below to switch examples between Staging and INI methods...

Show Staging

Show INI

```
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-replicator-7.1.2-81

shell> echo The staging USER is `tpm query staging| cut -d: -f1 | cut -d@ -f1`
The staging USER is tungsten

shell> echo The staging HOST is `tpm query staging| cut -d: -f1 | cut -d@ -f2`
The staging HOST is db1

shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-replicator-7.1.2-81

shell> ssh {STAGING_USER}@{STAGING_HOST}
shell> cd {STAGING_DIRECTORY}
```

```
shell> ./tools/tpm configure alpha \
    --svc-parallelization-type=disk \
    --channels=5
```

Run the tpm command to update the software with the Staging-based configuration:

```
shell> ./tools/tpm update
```

For information about making updates when using a Staging-method deployment, please see Section 9.3.7, “Configuration Changes from a Staging Directory”.

```
[alpha]
...
svc-parallelization-type=disk
channels=5
```

Run the tpm command to update the software with the INI-based configuration:

```
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-replicator-7.1.2-81

shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-replicator-7.1.2-81

shell> cd {STAGING_DIRECTORY}

shell> ./tools/tpm update
```

For information about making updates when using an INI file, please see Section 9.4.4, “Configuration Changes with an INI file”.

## 5.5.6. Monitoring Parallel Replication

Basic monitoring of a parallel deployment can be performed using the techniques in Chapter 7, *Operations Guide*. Specific operations for parallel replication are provided in the following sections.

### 5.5.6.1. Useful Commands for Parallel Monitoring Replication

The replicator has several helpful commands for tracking replication performance:

| Command | Description |
|---|---|
| trepctl status | Shows basic variables including overall latency of Replica and number of apply channels |
| trepctl status -name shards | Shows the number of transactions for each shard |
| trepctl status -name stores | Shows the configuration and internal counters for stores between tasks |
| trepctl status -name tasks | Shows the number of transactions (events) and latency for each independent task in the replicator pipeline |

### 5.5.6.2. Parallel Replication and Applied Latency On Replicas

The trepctl status appliedLastSeqno parameter shows the sequence number of the last transaction committed. Here is an example from a Replica with 5 channels enabled.

```
shell> trepctl status
Processing status command...
NAME                     VALUE
----                     -----
appliedLastEventId     : mysql-bin.000211:0000000020094456;0
appliedLastSeqno       : 78021
appliedLatency         : 0.216
channels               : 5
...
Finished status command...
```

When parallel apply is enabled, the meaning of `appliedLastSeqno` changes. It is the minimum recovery position across apply channels, which means it is the position where channels restart in the event of a failure. This number is quite conservative and may make replication appear to be further behind than it actually is.

- Busy channels mark their position in table `trep_commit_seqno` as they commit. These are up-to-date with the traffic on that channel, but channels have latency between those that have a lot of big transactions and those that are more lightly loaded.
- Inactive channels do not get any transactions, hence do not mark their position. Tungsten sends a control event across all channels so that they mark their commit position in `trep_commit_channel`. It is possible to see a delay of many seconds or even minutes in unloaded systems from the true state of the Replica because of idle channels not marking their position yet.

For systems with few transactions it is useful to lower the synchronization interval to a smaller number of transactions, for example 500. The following command shows how to adjust the synchronization interval after installation:

Click the link below to switch examples between Staging and INI methods...

Show Staging

Show INI

```
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-replicator-7.1.2-81

shell> echo The staging USER is `tpm query staging| cut -d: -f1 | cut -d@ -f1`
The staging USER is tungsten

shell> echo The staging HOST is `tpm query staging| cut -d: -f1 | cut -d@ -f2`
The staging HOST is db1

shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-replicator-7.1.2-81

shell> ssh {STAGING_USER}@{STAGING_HOST}
shell> cd {STAGING_DIRECTORY}
```

```
shell> ./tools/tpm configure alpha \
    --property=replicator.store.parallel-queue.syncInterval=500
```

Run the tpm command to update the software with the Staging-based configuration:

```
shell> ./tools/tpm update
```

For information about making updates when using a Staging-method deployment, please see Section 9.3.7, "Configuration Changes from a Staging Directory".

```
[alpha]
...
property=replicator.store.parallel-queue.syncInterval=500
```

Run the tpm command to update the software with the INI-based configuration:

```
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-replicator-7.1.2-81

shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-replicator-7.1.2-81

shell> cd {STAGING_DIRECTORY}

shell> ./tools/tpm update
```

For information about making updates when using an INI file, please see Section 9.4.4, "Configuration Changes with an INI file".

Note that there is a trade-off between the synchronization interval value and writes on the DBMS server. With the foregoing setting, all channels will write to the `trep_commit_seqno` table every 500 transactions. If there were 50 channels configured, this could lead to an increase in writes of up to 10%—each channel could end up adding an extra write to mark its position every 10 transactions. In busy systems it is therefore better to use a higher synchronization interval for this reason.

You can check the current synchronization interval by running the trepctl status -name stores command, as shown in the following example:

```
shell> trepctl status -name stores
Processing status command (stores)...
...
NAME                      VALUE
----                      -----
...
name                    : parallel-queue
...
storeClass              : com.continuent.tungsten.replicator.thl.THLParallelQueue
syncInterval            : 10000
Finished status command (stores)...
```

You can also force all channels to mark their current position by sending a heartbeat through using the trepctl heartbeat command.

## 5.5.6.3. Relative Latency

Relative latency is a trepctl status parameter. It indicates the latency since the last time the appliedSeqno advanced; for example:

```
shell> trepctl status
Processing status command...
NAME                     VALUE
----                     -----
appliedLastEventId     : mysql-bin.000211:0000000020094766;0
appliedLastSeqno       : 78022
appliedLatency         : 0.571
...
relativeLatency        : 8.944
Finished status command...
```

In this example the last transaction had a latency of .571 seconds from the time it committed on the Primary and committed 8.944 seconds ago. If relative latency increases significantly in a busy system, it may be a sign that replication is stalled. This is a good parameter to check in monitoring scripts.

## 5.5.6.4. Serialization Count

Serialization count refers to the number of transactions that the replicator has handled that cannot be applied in parallel because they involve dependencies across shards. For example, a transaction that spans multiple shards must serialize because it might cause cause an out-of-order update with respect to transactions that update a single shard only.

You can detect the number of transactions that have been serialized by looking at the *`serializationCount`* parameter using the trepctl status -name stores command. The following example shows a replicator that has processed 1512 transactions with 26 serialized.

```
shell> trepctl status -name stores
Processing status command (stores)...
...
```

```
NAME                     VALUE
----                     -----
criticalPartition       : -1
discardCount            : 0
estimatedOfflineInterval: 0.0
eventCount              : 1512
headSeqno               : 78022
maxOfflineInterval      : 5
maxSize                 : 10
name                    : parallel-queue
queues                  : 5
serializationCount      : 26
serialized              : false
...
Finished status command (stores)...
```

In this case 1.7% of transactions are serialized. Generally speaking you will lose benefits of parallel apply if more than 1-2% of transactions are serialized.

## 5.5.6.5. Maximum Offline Interval

The maximum offline interval (*maxOfflineInterval*) parameter controls the "distance" between the fastest and slowest channels when parallel apply is enabled. The replicator measures distance using the seconds between commit times of the last transaction processed on each channel. This time is roughly equivalent to the amount of time a replicator will require to go offline cleanly.

You can change the *maxOfflineInterval* as shown in the following example, the value is defined in seconds.

Click the link below to switch examples between Staging and INI methods...

Show Staging

Show INI

```
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-replicator-7.1.2-81

shell> echo The staging USER is `tpm query staging| cut -d: -f1 | cut -d@ -f1`
The staging USER is tungsten

shell> echo The staging HOST is `tpm query staging| cut -d: -f1 | cut -d@ -f2`
The staging HOST is db1

shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-replicator-7.1.2-81

shell> ssh {STAGING_USER}@{STAGING_HOST}
shell> cd {STAGING_DIRECTORY}
```

```
shell> ./tools/tpm configure alpha \
    --property=replicator.store.parallel-queue.maxOfflineInterval=30
```

Run the tpm command to update the software with the Staging-based configuration:

```
shell> ./tools/tpm update
```

For information about making updates when using a Staging-method deployment, please see Section 9.3.7, “Configuration Changes from a Staging Directory”.

```
[alpha]
...
property=replicator.store.parallel-queue.maxOfflineInterval=30
```

Run the tpm command to update the software with the INI-based configuration:

```
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-replicator-7.1.2-81

shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-replicator-7.1.2-81

shell> cd {STAGING_DIRECTORY}

shell> ./tools/tpm update
```

For information about making updates when using an INI file, please see Section 9.4.4, “Configuration Changes with an INI file”.

You can view the configured value as well as the estimate current value using the trepctl status -name stores command, as shown in yet another example:

```
shell> trepctl status -name stores
Processing status command (stores)...
NAME                      VALUE
----                      -----
...
estimatedOfflineInterval: 1.3
...
maxOfflineInterval      : 30
...
Finished status command (stores)...
```

## 5.5.6.6. Workload Distribution

Parallel apply works best when transactions are distributed evenly across shards and those shards are distributed evenly across available channels. You can monitor the distribution of transactions over shards using the trepctl status -name shards command. This command lists transaction counts for all shards, as shown in the following example.

```
shell> trepctl status -name shards
Processing status command (shards)...
...
NAME                VALUE
----                -----
appliedLastEventId: mysql-bin.000211:0000000020095076;0
appliedLastSeqno  : 78023
appliedLatency    : 0.255
eventCount        : 3523
shardId           : cust1
stage             : q-to-dbms
...
Finished status command (shards)...
```

If one or more shards have a very large *eventCount* value compared to the others, this is a sign that your transaction workload is poorly distributed across shards.

The listing of shards also offers a useful trick for finding serialized transactions. Shards that Tungsten Replicator cannot safely parallelize are assigned the dummy shard ID `#UNKNOWN`. Look for this shard to find the count of serialized transactions. The *appliedLastSeqno* for this shard gives the sequence number of the most recent serialized transaction. As the following example shows, you can then list the contents of the transaction to see why it serialized. In this case, the transaction affected tables in different schemas.

```
shell> trepctl status -name shards
Processing status command (shards)...
NAME                VALUE
----                -----
appliedLastEventId: mysql-bin.000211:0000000020095529;0
appliedLastSeqno  : 78026
appliedLatency    : 0.558
eventCount        : 26
shardId           : #UNKNOWN
stage             : q-to-dbms
...
Finished status command (shards)...
shell> thl list -seqno 78026
SEQ# = 78026 / FRAG# = 0 (last frag)
- TIME = 2013-01-17 22:29:42.0
- EPOCH# = 1
- EVENTID = mysql-bin.000211:0000000020095529;0
- SOURCEID = logos1
- METADATA = [mysql_server_id=1;service=percona;shard=#UNKNOWN]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [##charset = ISO8859_1, autocommit = 1, sql_auto_is_null = 0, »
    foreign_key_checks = 1, unique_checks = 1, sql_mode = '', character_set_client = 8, »
    collation_connection = 8, collation_server = 33]
- SCHEMA =
- SQL(0) = insert into mats_0.foo values(1) /* ___SERVICE___ = [percona] */
- OPTIONS = [##charset = ISO8859_1, autocommit = 1, sql_auto_is_null = 0, »
    foreign_key_checks = 1, unique_checks = 1, sql_mode = '', character_set_client = 8, »
    collation_connection = 8, collation_server = 33]
- SQL(1) = insert into mats_1.foo values(1)
```

The replicator normally distributes shards evenly across channels. As each new shard appears, it is assigned to the next channel number, which then rotates back to 0 once the maximum number has been assigned. If the shards have uneven transaction distributions, this may lead to an uneven number of transactions on the channels. To check, use the trepctl status -name tasks and look for tasks belonging to the `q-to-dbms` stage.

```
shell> trepctl status -name tasks
Processing status command (tasks)...
...
NAME                VALUE
----                -----
```

```
appliedLastEventId: mysql-bin.000211:0000000020095076;0
appliedLastSeqno  : 78023
appliedLatency    : 0.248
applyTime         : 0.003
averageBlockSize  : 2.520
cancelled         : false
currentLastEventId: mysql-bin.000211:0000000020095076;0
currentLastFragno : 0
currentLastSeqno  : 78023
eventCount        : 5302
extractTime       : 274.907
filterTime        : 0.0
otherTime         : 0.0
stage             : q-to-dbms
state             : extract
taskId            : 0
...
Finished status command (tasks)...
```

If you see one or more channels that have a very high *eventCount*, consider either assigning shards explicitly to channels or redistributing the workload in your application to get better performance.

## 5.5.7. Controlling Assignment of Shards to Channels

Tungsten Replicator by default assigns channels using a round robin algorithm that assigns each new shard to the next available channel. The current shard assignments are tracked in table `trep_shard_channel` in the Tungsten catalog schema for the replication service.

For example, if you have 2 channels enabled and Tungsten processes three different shards, you might end up with a shard assignment like the following:

```
foo => channel 0
bar => channel 1
foobar => channel 0
```

This algorithm generally gives the best results for most installations and is crash-safe, since the contents of the `trep_shard_channel` table persist if either the DBMS or the replicator fails.

It is possible to override the default assignment by updating the `shard.list` file found in the `tungsten-replicator/conf` directory. This file normally looks like the following:

```
# SHARD MAP FILE.
# This file contains shard handling rules used in the ShardListPartitioner
# class for parallel replication.  If unchanged shards will be hashed across
# available partitions.

# You can assign shards explicitly using a shard name match, where the form
# is <db>=<partition>.
#common1=0
#common2=0
#db1=1
#db2=2
#db3=3

# Default partition for shards that do not match explicit name.
# Permissible values are either a partition number or -1, in which
# case values are hashed across available partitions.  (-1 is the
# default.
#(*)=-1

# Comma-separated list of shards that require critical section to run.
# A "critical section" means that these events are single-threaded to
# ensure that all dependencies are met.
#(critical)=common1,common2

# Method for channel hash assignments.  Allowed values are round-robin and
# string-hash.
(hash-method)=round-robin
```

You can update the shard.list file to do three types of custom overrides.

1. Change the hashing method for channel assignments. Round-robin uses the `trep_shard_channel` table. The string-hash method just hashes the shard name.
2. Assign shards to explicit channels. Add lines of the form *shard=channel* to the file as shown by the commented-out entries.
3. Define critical shards. These are shards that must be processed in serial fashion. For example if you have a sharded application that has a single global shard with reference information, you can declare the global shard to be critical. This helps avoid applications seeing out of order information.

Changes to shard.list must be made with care. The same cautions apply here as for changing the number of channels or the parallelization type. For subscription customers we strongly recommend conferring with Continuent Support before making changes.

## 5.5.8. Disk vs. Memory Parallel Queues

Channels receive transactions through a special type of queue, known as a parallel queue. Tungsten offers two implementations of parallel queues, which vary in their performance as well as the requirements they may place on hosts that operate parallel apply. You choose the type of queue to enable using the `--svc-parallelization-type` [425] option.

### Warning

Do not change the parallel queue type without setting the replicator offline cleanly. See the procedure later in this page for more information.

Disk Parallel Queue (*disk* option)

A disk parallel queue uses a set of independent threads to read from the Transaction History Log and feed short in-memory queues used by channels. Disk queues have the advantage that they minimize memory required by Java. They also allow channels to operate some distance apart, which improves throughput. For instance, one channel may apply a transaction that committed 2 minutes before the transaction another channel is applying. This separation keeps a single slow transaction from blocking all channels.

Disk queues minimize memory consumption of the Java VM but to function efficiently they do require pages from the Operating System page cache. This is because the channels each independently read from the Transaction History Log. As long as the channels are close together the storage pages tend to be present in the Operating System page cache for all threads but the first, resulting in very fast reads. If channels become widely separated, for example due to a high *maxOfflineInterval* value, or the host has insufficient free memory, disk queues may operate slowly or impact other processes that require memory.

Memory Parallel Queue (*memory* option)

A memory parallel queue uses a set of in-memory queues to hold transactions. One stage reads from the Transaction History Log and distributes transactions across the queues. The channels each read from one of the queues. In-memory queues have the advantage that they do not need extra threads to operate, hence reduce the amount of CPU processing required by the replicator.

When you use in-memory queues you must set the maxSize property on the queue to a relatively large value. This value sets the total number of transaction fragments that may be in the parallel queue at any given time. If the queue hits this value, it does not accept further transaction fragments until existing fragments are processed. For best performance it is often necessary to use a relatively large number, for example 10,000 or greater.

The following example shows how to set the maxSize property after installation. This value can be changed at any time and does not require the replicator to go offline cleanly:

Click the link below to switch examples between Staging and INI methods...

Show Staging

Show INI

```
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-replicator-7.1.2-81

shell> echo The staging USER is `tpm query staging| cut -d: -f1 | cut -d@ -f1`
The staging USER is tungsten

shell> echo The staging HOST is `tpm query staging| cut -d: -f1 | cut -d@ -f2`
The staging HOST is db1

shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-replicator-7.1.2-81

shell> ssh {STAGING_USER}@{STAGING_HOST}
shell> cd {STAGING_DIRECTORY}
```

```
shell> ./tools/tpm configure alpha \
    --property=replicator.store.parallel-queue.maxSize=10000
```

Run the tpm command to update the software with the Staging-based configuration:

```
shell> ./tools/tpm update
```

For information about making updates when using a Staging-method deployment, please see Section 9.3.7, “Configuration Changes from a Staging Directory”.

```
[alpha]
...
```

```
property=replicator.store.parallel-queue.maxSize=10000
```

Run the tpm command to update the software with the INI-based configuration:

```
shell> tpm query staging
tungsten@db1:/opt/continuent/software/tungsten-replicator-7.1.2-81

shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-replicator-7.1.2-81

shell> cd {STAGING_DIRECTORY}

shell> ./tools/tpm update
```

For information about making updates when using an INI file, please see Section 9.4.4, “Configuration Changes with an INI file”.

You may need to increase the Java VM heap size when you increase the parallel queue maximum size. Use the `--java-mem-size` [410] option on the tpm command for this purpose or edit the Replicator `wrapper.conf` file directly.

### Warning

Memory queues are not recommended for production use at this time. Use disk queues.

# 5.6. Batch Loading for Data Warehouses

Tungsten Replicator normally applies SQL changes to Targets by constructing SQL statements and executing in the exact order that transactions appear in the Tungsten History Log (THL). This works well for OLTP databases like MySQL, Oracle, and MongoDB. However, it is a poor approach for data warehouses.

Data warehouse products like Vertica or Redshift load very slowly through JDBC interfaces (50 times slower or even more compared to MySQL). Instead, such databases supply batch loading commands that upload data in parallel. For instance Vertica uses the `COPY` command.

Tungsten Replicator has a batch applier named SimpleBatchApplier that groups transactions and then loads data. This is known as "batch apply." You can configure Tungsten to load 10s of thousands of transactions at once using template that apply the correct commands for your chosen data warehouse.

While we use the term *batch apply* Tungsten is not batch-oriented in the sense of traditional Extract/Transfer/Load tools, which may run only a small number of batches a day. Tungsten builds batches automatically as transactions arrive in the log. The mechanism is designed to be self-adjusting. If small transaction batches cause loading to be slower, Tungsten will automatically tend to adjust the batch size upwards until it no longer lags during loading.

## 5.6.1. How It Works

The batch applier loads data into the Target DBMS using CSV files and appropriate load commands like `LOAD DATA INFILE` or `COPY`. Here is the basic algorithm.

While executing within a commit block, we write incoming transactions into open CSV files written by the class `CsvWriter`. There is one CSV file per database table. The following sample shows typical contents.

```
"I","84900","1","2016-03-11 20:51:10.000","986","http://www.continent.com/software"
"D","84901","2","2016-03-11 20:51:10.000","143",null
"I","84901","3","2016-03-11 20:51:10.000","143","http://www.microsoft.com"
```

Tungsten adds four extra column values to each line of CSV output.

| Column | Description |
|---|---|
| `opcode` | A transaction code that has the value "I" for insert and "D" for delete. Other types are available. |
| `seqno` | The Tungsten transaction sequence number |
| `row_id` | A line number that starts with 1 and increments by 1 for each new row |
| `timestamp` | The commit timestamp, i.e. the origin timestamp of the committed statement that generated the row information. |

Different update types are handled as follows:

- Each insert generates a single row containing all values in the row with an "I" opcode.
- Each delete generates a single row with the key and a "D" opcode. Non-key fields are null.
- Each update results in a delete with the row key followed by an insert.
- Statements are ignored. If you want DDL you need to put it in yourself.

Tungsten writes each row update into the corresponding CSV file for the SQL. At commit time the following steps occur:

1. Flush and close each CSV file. This ensures that if there is a failure the files are fully visible in storage.
2. For each table execute a merge script to move the data from CSV into the data warehouse. This script varies depending on the data warehouse type or even for specific application. It generally consists of a sequence of operating system commands, load commands like `COPY` or `LOAD DATA INFILE` to load in the CSV data, and ordinary SQL commands to move/massage data.
3. When all tables are loaded, issue a single commit on the SQL connection.

The main requirement of merge scripts is that they must ensure rows load and that delete and insert operations apply in the correct order. Tungsten includes load scripts for MySQL and Vertica that do this automatically.

It is common to use staging tables to help load data. These are described in more detail in a later section.

## 5.6.2. Important Limitations

Tungsten currently has some important limitations for batch loading, namely:

1. Primary keys must be a single column only. Tungsten does not handle multi-column keys.
2. Binary data is not certified and may cause problems when converted to CSV as it will be converted to Unicode.

These limitations will be relaxed in future releases.

## 5.6.3. Batch Applier Setup

Here is how to set up on MySQL. For more information on specific data warehouse types, refer to Chapter 2, *Deployment Overview*.

1. Enable row replication on the MySQL Source using set global `binlog_format=row` or by updating `my.cnf`.
2. Ensure that you are operating using GMT throughout your source and target database.
3. Install using the `--batch-enabled=true` [397] option. Here's a typical vertica applier configuration, taken from Section 4.3, “Deploying the Vertica Applier” :.

   Show Staging

   Show INI

```
shell> ./tools/tpm configure defaults \
    --reset \
    --user=tungsten \
    --install-directory=/opt/continuent \
    --profile-script=~/.bash_profile \
    --skip-validation-check=HostsFileCheck \
    --skip-validation-check=InstallerMasterSlaveCheck \
    --rest-api-admin-user=apiuser \
    --rest-api-admin-pass=secret

shell> ./tools/tpm configure alpha \
    --topology=master-slave \
    --master=sourcehost \
    --members=localhost \
    --datasource-type=vertica \
    --replication-user=dbadmin \
    --replication-password=password \
    --vertica-dbname=dev \
    --batch-enabled=true \
    --batch-load-template=vertica6 \
    --batch-load-language=js \
    --replication-port=5433 \
    --svc-applier-filters=dropstatementdata \
    --svc-applier-block-commit-interval=30s \
    --svc-applier-block-commit-size=25000 \
    --disable-relay-logs=true
```

```
shell> vi /etc/tungsten/tungsten.ini
```

```
[defaults]
user=tungsten
install-directory=/opt/continuent
profile-script=~/.bash_profile
skip-validation-check=HostsFileCheck
skip-validation-check=InstallerMasterSlaveCheck
rest-api-admin-user=apiuser
```

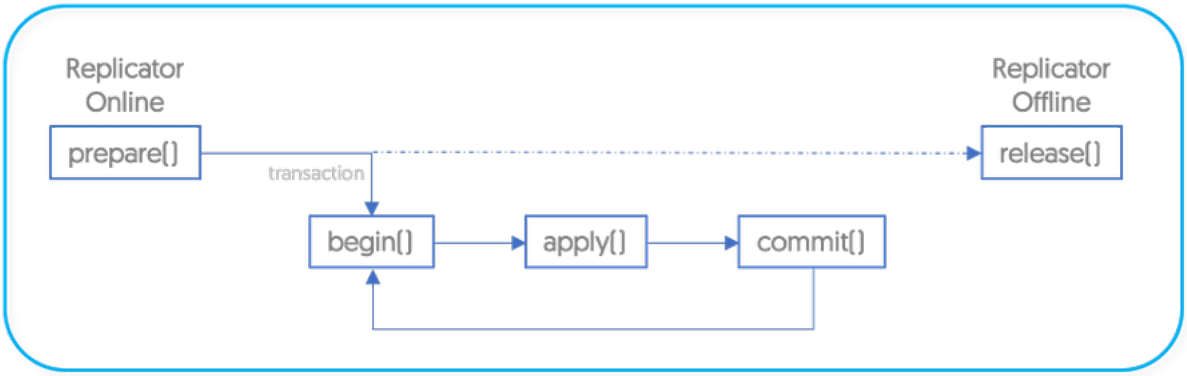
```
rest-api-admin-pass=secret

[alpha]
topology=master-slave
master=sourcehost
members=localhost
datasource-type=vertica
replication-user=dbadmin
replication-password=password
vertica-dbname=dev
batch-enabled=true
batch-load-template=vertica6
batch-load-language=js
replication-port=5433
svc-applier-filters=dropstatementdata
svc-applier-block-commit-interval=30s
svc-applier-block-commit-size=25000
disable-relay-logs=true
```

## 5.6.4. JavaScript Batchloader Scripts

The JavaScript batchloader enables data to be loaded into datawarehouse and other targets through a simplified JavaScript command script. The script implements specific functions for specification stages for the apply process, from preparation to commit, allowing for internal data, external commands, and other operations to be executed in sequence.

The actual loading process works through the specification of a JavaScript batchload script that defines what operations to perform during each stage of the batchloading process. These mirror the basic steps in the operation of applying the data that is being batchloaded, as shown in Figure 5.3, “Batchloading: JavaScript”.

Figure 5.3. Batchloading: JavaScript




To summarize:

- prepare() is called when the replicator goes online
- begin() is called before a single transaction starts
- apply() is called to copy and load the raw CSV data
- commit() is called after the raw data has been loaded
- release() is called when the replicator goes offline

### 5.6.4.1. JavaScript Batchloader with Parallel Apply

The JavaScript batchloader can be used with parallel apply to enable multiple threads to be generated and apply data to the target database. This can be useful in datawarehouse environments where simultaneous loading (and commit) enables effective application of multiple table data into the datawarehouse.

- The defined JavaScript methods like prepare, begin, commit, and release are called independently for each environment. This means that you should ensure actions in these methods do not conflict with each other.
- CSV files are divided across the scripts. If there is a large number of files that all take about the same time to load and there are three threads (parallelization=3), each individual load script will see about a third of the files. You should therefore not code assumptions that you have seen all tables or CSV files in a single script.

- Parallel load script is only recommended for data sources like Hadoop that are idempotent. When applying to a data source that is non-idempotent (for example MySQL or potentially Vertica) you should just use a single thread.

## 5.6.5. Staging Tables

Staging tables are intermediate tables that help with data loading. There are different usage patterns for staging tables.

### 5.6.5.1. Staging Table Names

Tungsten assumes that staging tables, if present, follow certain conventions for naming and provides a number of configuration properties for generating staging table names that match the base tables in the data warehouse without colliding with them.

| Property | Description |
|---|---|
| *stageColumnPrefix* | Prefix for seqno, row_id, and opcode columns generated by Tungsten |
| *stageTablePrefix* | Prefix for stage table name |
| *stageSchemaPrefix* | Prefix for the schema in which the stage tables reside |

These values are set in the static properties file that defines the replication service. They can be set at install time using `--property` [368] options. The following example shows typical values from a service properties file.

```
replicator.applier.dbms.stageColumnPrefix=tungsten_
replicator.applier.dbms.stageTablePrefix=stage_xxx_
replicator.applier.dbms.stageSchemaPrefix=load_
```

If your data warehouse contains a table named `foo` in schema `bar`, these properties would result in a staging table name of `load_bar.stage_xxx_foo` for the staging table. The Tungsten generated column containing the `seqno`, if present, would be named `tungsten_seqno`.

#### Note

Staging tables are by default in the same schema as the table they update. You can put them in a different schema using the *stageSchemaPrefix* property as shown in the example.

### 5.6.5.2. Whole Record Staging

Whole record staging loads the entire CSV file into an identical table, then runs queries to apply rows to the base table or tables in the data warehouse. One of the strengths of whole record staging is that it allows you to construct a merge script that can handle any combination of `INSERT`, `UPDATE`, or `DELETE` operations. A weakness is that whole record staging can result in sub-optimal I/O for workloads that consist mostly of `INSERT` operations.

For example, suppose we have a base table created by the following `CREATE TABLE` command:

```
CREATE TABLE `mydata` (
`id` int(11) NOT NULL,
`f_data` float DEFAULT NULL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

A whole record staging table would look as follows.

```
CREATE TABLE `stage_xxx_croc_mydata` (
`tungsten_opcode` char(1) DEFAULT NULL,
`tungsten_seqno` int(11) DEFAULT NULL,
`tungsten_row_id` int(11) DEFAULT NULL,
`id` int(11) NOT NULL,
`f_data` float DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Note that this table does not have a primary key defined. Most data warehouses do not use primary keys and many of them do not even permit it in the create table syntax.

Note also that the non-primary columns must permit nulls. This is required for deletes, which contain only the Tungsten generated columns plus the primary key.

### 5.6.5.3. Delete Key Staging

Another approach is to load `INSERT` rows directly into the base data warehouse tables without staging. All you need to stage is the keys for deleted records. This reduces I/O considerably for workloads that have mostly inserts. The downside is that it may require introduce ordering dependencies between `DELETE` and `INSERT` operations that require special handling by upstream applications to generate transactions that will load without conflicts.

Delete key staging tables can be as simple as the follow example:

```
CREATE TABLE `stage_xxx_croc_mydata` (
`id` int(11) NOT NULL,
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

### 5.6.5.4. Staging Table Generation

Tungsten does not generate staging tables automatically. Creation of staging tables is the responsibility of users, but using the ddlscan tool with the right template can be simplified.

## 5.6.6. Character Sets

Character sets are a headache in batch loading because all updates are written and read from CSV files, which can result in invalid transactions along the replication path. Such problems are very difficult to debug. Here are some tips to improve chances of happy replicating.

- Use UTF8 character sets consistently for all string and text data.
- Force Tungsten to convert data to Unicode rather than transferring strings:

```
shell> mysql-use-bytes-for-string=false
```

- When starting the replicator for MySQL replication, include the following option:

```
shell> java-file-encoding=UTF8
```

## 5.6.7. Supported CSV Formats

Tungsten Replicator supports a number of CSV formats that can and should be used with specific heterogeneous environments when using the batch loading process, or generating CSV files in general for testing or loading.

A number of standard types are included, and the use of these standard types when generating CSV is controlled by the replicator.datasource.global.csvType property. Depending on the configured target, the corresponding type will be configured automatically. For example, if you configure a Vertica deployment, the replicator will be configured to default to the Vertica style CSV format.

> **Warning**
>
> Using the wrong CSV format with a given target may break replication. You should always use the appropriate CSV format for the defined target.

Table 5.1. Continuent Tungsten Directory Structure

| Format | Field Separator | Record Separator | Escape Sequence | Escaped Characters | Null Policy | Null Value | Show Headers | Use Quotes | Quote String | Suppressed Characters |
|---|---|---|---|---|---|---|---|---|---|---|
| hive | \u0001 | \n | \\ | \u0001\\ | Use Null Value | \\N | false | false | | \n\r |
| mysql | , | \n | \\ | \\ | Use Null Value | \\N | false | true | \" | |
| oracle | , | \n | \\ | \\ | Use Null Value | \\N | false | true | \" | |
| vertica | , | \n | \\ | \\ | Skip Value | | false | true | \" | \n |
| redshift | , | \n | \" | | Skip Value | | false | true | \" | \n |

In addition to the standardised types, the replicator.datasource.global.csvType property can be set to `custom`, in which case the following configurable values are used instead:

- replicator.datasource.global.csv.fieldSeparator — the character used to separate fields, such as `,` (comma).
- replicator.datasource.global.csv.RecordSeparator — the character used to separate records, such as the newline character.
- replicator.datasource.global.csv.nullValue — the value to use for NULL (empty) values.
- replicator.datasource.global.csv.useQuotes — whether to use quotes to encapsulate field values (specified using `true` or `false`).
- replicator.datasource.global.csv.useHeaders — whether to include the column headers in the generated CSV (specified using `true` or `false`).

## 5.6.8. Columns in Generated CSV Files

The CSV generated when using the batch loading process creates a number of special columns that are designed to hold the appropriate information for loading the staging data into the target system.

There are four fields supported:

- `opcode` — The operation code, a one- or two-letter code indicating the operation type. For more information on the supported codes, see Section 5.6.9, "Batchloading Opcodes".
- `seqno` — Contains the current THL event (sequence) number for the row data being loaded. The sequence number generated is specific to the THL event number.
- `row_id` — Contains a unique row ID (a monotonically incrementing number) which is unique to this CSV file for the table data being loaded. This can be useful for systems where the sequence number alone is not enough to identify an incoming row, even with the incoming primary key information.
- `commit_timestamp` — the timestamp of when the data was originally committed by the source database, taken from the *TIME* [550] within the THL event.
- `service` — the service name of the replicator service that performed the loading and generated the CSV. This field is not enabled by default, but is provided to allow for data concentration into a BigData target while enabling identification of the source service and/or database that generated the data.

These fields are placed before the actual data for the corresponding table, for example, with the default setting, the following CSV is generated, the last three columns are specific to the table data:

```
"I","74","1","2017-05-26 13:00:11.000","655337","Dr No","kat"
```

The configuration of the list of fields, and the order in which they appear, is controlled by the replicator.applier.dbms.stageColumnNames property. By default, all four fields, in the order shown above, are used:

```
replicator.applier.dbms.stageColumnNames=opcode,seqno,row_id,commit_timestamp
```

The actual names used (and passed to the JavaScript environment) are also controlled by another property, replicator.applier.dbms.stageColumnPrefix. This value is prepended to each column within the JS environment, and expected by the various tools. For example, with the default `tungsten_` the true name for the `opcode` is `tungsten_opcode`.

### Warning

Modifying the list of fields generated by the CSV writer may stop batchloading from working. Unless otherwise noted, the default batchloading scripts all expect to see the default four columns (`opcode`, `seqno`, `row_id` and `commit_timestamp`.

## 5.6.9. Batchloading Opcodes

The batchloading an CSV generation process use the `opcode` value to specify the operation type for each row. The default mode is to use only the `I` and `D` codes for inserts and deletes respectively, with an update being represented as two rows, one a delete and the other an insert of the new information.

This behavior can be altered to denote updates with a `U` character, with the row containing the updated information. To enable this mode, set the replicator.applier.dbms.useUpdateOpcode to `true`.

It is also possible to identify situations where the incoming row data indicates a delete operation that resulted from an update (for example, in a cascade or related column), and an insert from an update. When this mode is enable, the `opcode` becomes a two-character value or `UD` and `UI` respectively. To enable this option, set the replicator.applier.dbms.distinguishUpdates property to `true`.

### Warning

Changing the default opcode modes may cause replication to fail. The default JavaScript batchloading scripts expect the default `I` and `D` notation with updated implied through a delete and insert operation.

## 5.6.10. Time Zones

Time zones are another headache when using batch loading. For best results applications should standardize on a single time zone, preferably UTC, and use this consistently for all data. To ensure the Java VM outputs time data correctly to CSV files, you must set the JVM time zone to be the same as the standard time zone for your data. Here is the JVM setting in wrapper.conf:

```
# To ensure consistent handling of dates in heterogeneous and batch replication
# you should set the JVM timezone explicitly.  Otherwise the JVM will default
# to the platform time, which can result in unpredictable behavior when
```

```
# applying date values to Targets.  GMT is recommended to avoid inconsistencies.
wrapper.java.additional.5=-Duser.timezone=GMT
```

#### Note

> Beware that MySQL has two very similar data types: `TIMESTAMP` and `DATETIME`. Timestamps are stored in UTC and convert back to local time on display. Datetimes by contrast do not convert back to local time. If you mix timezones and use both data types your time values will be inconsistent on loading.

## 5.6.11. Batch Loading into MySQL

#### Note

> All the features discussed in this section are only available from version 6.1.15 of Tungsten Replicator

There are occasions where Batch loading into MySQL may benefit your use case, such as loading large data warehouse environments, or where real-time replication isn't as critical.

A number of specific properties are available for MySQL targets, these are discussed below.

### 5.6.11.1. Configuring as an Offboard Batch Applier

By Default, when loading into MySQL using the Batch Applier, the process executes `LOAD DATA INFILE` statements to load the CSV files into the database.

If you wish to install the applier on a remote host, this action would typically fail, therefore you need to enable the following property in the configuration:

```
property=replicator.applier.dbms.useLoadDataLocalInfile=true
```

### 5.6.11.2. Drop Delete Statements

Tungsten Replicator includes a number of useful filters, such as the ability to drop certain DML statements on a schema or table level.

If you wish to drop such statements on a per object basis, then you should continue to use the skipbyevent filter, however if you want to drop ALL `DELETE` DML, then you can enable the following property:

```
property=replicator.applier.dbms.skipDeletes=true
```

#### Warning

> By dropping deletes, you will then subsequently expose yourself to errors should rows be reinserted later with the same Primary or Unique Key values. Typically, this feature would be only enabled when you plan to capture and log key violations. See Section 5.6.11.6, “Log rows violating Primary/Unique Keys” for more information.

### 5.6.11.3. Configure CHARSET to use on Load

If you wish to specify a different CHARSET to be used when the data is being loaded into the target database, this can be set using the following property, for example:

```
property=replicator.applier.dbms.loadCharset=utf8mb4
```

### 5.6.11.4. Allow DDL Statements to execute

Typically, the batch loader is used for heterogeneous targets, and therefore by default DDL statements will be dropped. However, when applying into MySQL the DDL statements would be valid and can therefore be executed.

To enable this, you should set the following property:

```
property=replicator.applier.dbms.applyStatements=true
```

#### Warning

> Any changes to existing tables, or creation of new tables, will only apply to the main base table. You will still need to manually make changes to the relevant staging and error tables (if used)

### 5.6.11.5. Disable Foreign Keys during load

If you use a lot of foreign keys in your target database, due to the nature of batch loading, this could cause errors when tables may not be loaded in sequence meaning child/parent keys may only be validated after a complete transaction load.

To prevent this from happening, you can enable the property below which will force the batch loader to temporarily disable foreign key checks until after the full transaction has been loaded.

```
property=replicator.applier.dbms.disableForeignKeys=true
```

### 5.6.11.6. Log rows violating Primary/Unique Keys

To prevent the replicator erroring on primary or unique key violations, you can instruct the replicator to log the offending rows in an error table, which will allow you to manually process the rows afterwards.

This is especially useful when you are dropping DELETE statements from the apply process

The following properties can be set to enable this:

```
property=replicator.applier.dbms.useUpdateOpcode=true
property=replicator.applier.dbms.batchLogDuplicateRows=true
```

By default, this feature will only check against `PRIMARY KEYS`, if you wish to also check against `UNIQUE` keys, you will need the additional property:

```
property=replicator.applier.dbms.fetchKeysFromDatabase=true
```

By default, the error rows will be logged into tables called `error_xxx_origTableName`.

These table will need precreating in the same way that you create the Staging tablesusing ddlscan, but supplying the table prefix, for example:

```
shell> ddlscan -db hr -template ddl-mysql-staging.vm -opt tablePrefix error_xxx_
```

You can choose a different prefix if you wish, by replacing the error_xxx with you choice in the above ddlscan statement. If you choose to do this, you will also need to supply the new prefix in your configuration using the following property:

```
property=replicator.applier.dbms.errorTablePrefix=your-prefix-here_
```

#### Warning

> If you are loading 10's of thousands of rows per transaction, and your target tables are very large, this process could slow down the apply process as the applier will first need to ensure the row being inserted does not violate any keys. The use of this feature should be fully tested in a load test environment and the risks fully understood before using in production.

## 5.6.12. Data File Partitioning

By default, the CSV files generated as part of the batchloading process are named according to the schema name, table name, and the starting transaction sequence number that generated the data in the file. For example, the table `orders` within the schema `sales` generating the transaction information from sequence numbers 110 through 145 would have the name `sales-orders-110.csv`.

Because the size of the files can be quite large, and because within different target environments (particularly Hadoop or when uploading to S3) the speed with which the data can be uploaded or organised within the target can be critical, the files can also be partitioned. This splits up the files generated by a chosen value such as the commit time or data value.

The primary solution for partitioning is to the DateTime partitioner, which then uses a configurable date time value from the internal data structure to act as the basis for the information.

To enable date-based partitioning, you must specify the properties during your configuration:

```
replicator.applier.dbms.partitionBy=tungsten_commit_timestamp
replicator.applier.dbms.partitionByClass=com.continuent.tungsten.replicator.applier.batch.DateTimeValuePartitioner
replicator.applier.dbms.partitionByFormat=yyyy-MM-dd-HH
```

The above sets the use fo the `tungsten_commit_timestamp` field generated by the batchload CSV system as the basis of the value. The format specification is then used to specify the format of the data which will be embedded into the file. The data formatter uses the Java date format strings, and you can use one or more of the following values:

- `YY`

  Year as two digit number

- `yyyy`

  Year as four digit number

- `MM`

Month with leading zero

- `dd`

  Day with leading zero

- `HH`

  Hour in 24 hour format with leading zero

- `mm`

  Minute with leading zero

- `ss`

  Seconds with leading zero

For example, setting `yyyy-MM-dd-HH` (the default), the name of the CSV file will be `orders-sales-2018-04-03-12-199.csv`. Note that the THL sequence number is still embedded in the filename (as the last item), as is the schema and table name.

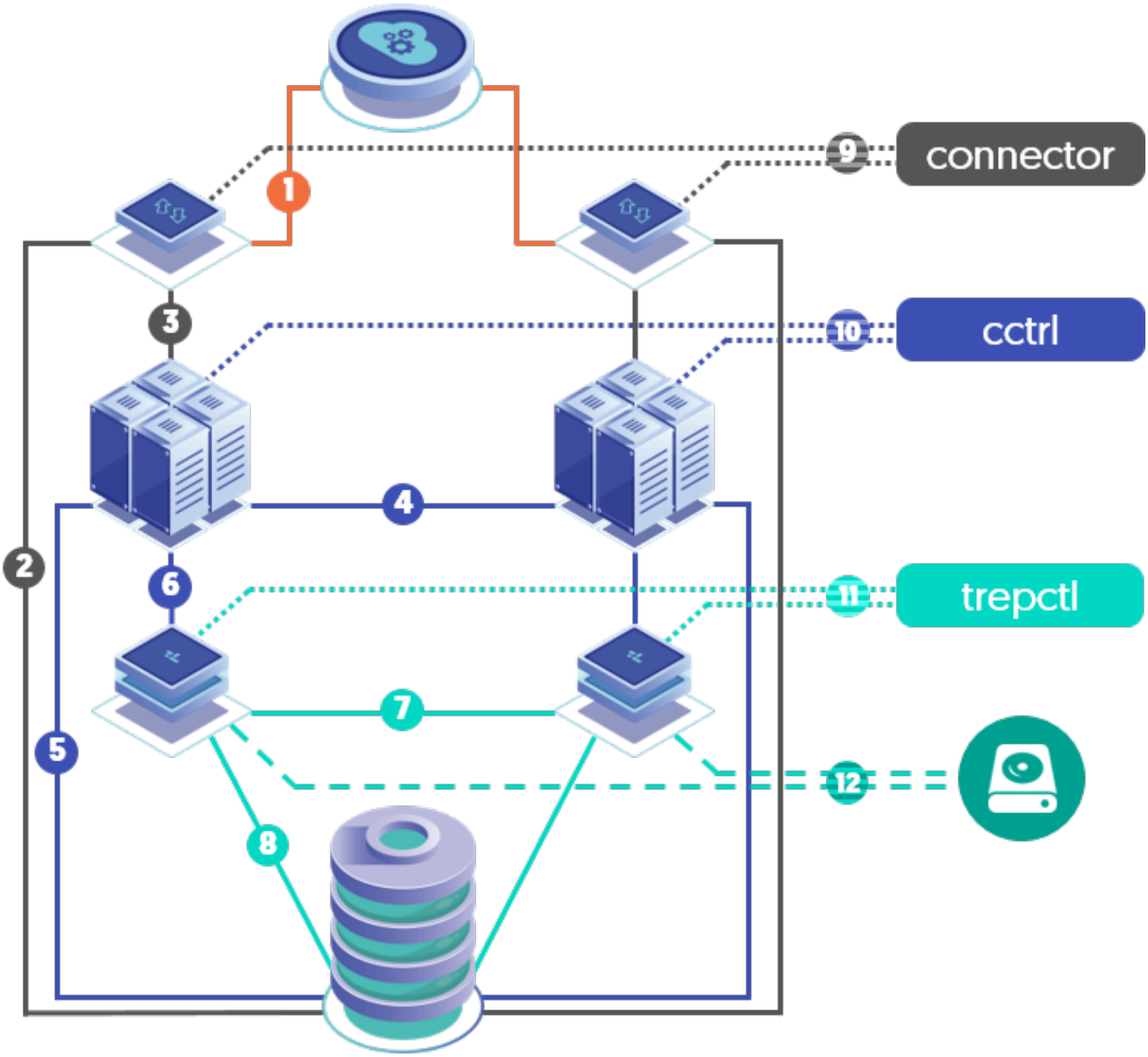
Files generated will automatically be split by the configured value, but remember that the commit timestamp will be consistent for an individual transaction, so data will never be split across multiple files for a single transaction even if it takes time for the CSV file to be written, the key is the commit timestamp from the source database for the entire transaction that corresponds to the sequence number.

# Chapter 6. Deployment: Security

- Authentication between command-line tools (trepctl), and between background services.
- SSL/TLS between command-line tools and background services.
- SSL/TLS between Tungsten Replicator and datasources.
- SSL for all API calls.
- File permissions and access by all components.

The following graphic provides a visual representation of the various communication channels which may be encrypted.

Figure 6.1. Security Internals: Cluster Communication Channels




For the key to the above diagram, please see Section 9.5.15, "tpm report Command".

If you are using a single staging directory to handle your complete installation, tpm will automatically create the necessary certificates for you. If you are using an INI based installation, then the installation process will create the certificates for you, however you will need to manually sync them between hosts prior to starting the various components.

It is assumed that your underlying database has SSL enabled and the certificates are available. If you need, and want, this level of security enabling, you can refer to Section 6.10.1, "Enabling Database SSL" for the steps required.

Additionally, if you are configuring heterogeneous replication there will additional manual steps required to ensure SSL communication to you chosen target database.

> **Important**
>
> Due to a known issue in earlier Java revisions that may cause performance degradation with client connections, it is strongly advised that you ensure your Java version is one of the following MINIMUM releases before enabling SSL:
>
> - Oracle JRE 8 Build 261
> - OpenJDK 8 Build 222

# 6.1. Enabling Security

By default, security is enabled for new installations.

Security can be enabled/disabled by adding the `disable-security-controls` [404] option to the configuration.

If this property is not supplied, or set to false, then security will be enabled. If set to true, then security will be disabled.

Enabling security through this single option, has the same effect as adding:

- `--file-protection-level=0027` [408]
- `--rmi-ssl=true` [407]
- `--thl-ssl=true` [408]
- `--rmi-authentication=true` [407]
- `--datasource-enable-ssl=true` [400]
- `--replicator-rest-api-ssl=true` [421]

> **Important**
>
> If you are enabling to-the-database encryption, you must ensure this has been enabled in your database and the relevant certificates are available first. See Section 6.10.1, "Enabling Database SSL" for steps.

> **Important**
>
> Installing from a staging host will automatically generate certificates and configuration for a secured installation. No further changes or actions are required.
>
> For INI-based installations, there are additional steps required to copy the needed certificate files to all of the nodes. Please see Section 6.1.2, "Enabling Security using the INI Method" for details.

## 6.1.1. Enabling Security using the Staging Method

Security will be enabled during initial install by default, should you choose to disable at install, then these steps will guide you in the process to enable as part of a post-install update

Enabled During Install

As mentioned, security is enabled by default. This is controlled by the `--disable-security-controls=false` [404]. If not supplied, the default is false. You can choose to specify this in your configuration for transparency if you wish.

```
shell> tools/tpm configure defaults --disable-security-controls=false \
[...the rest of the configuration options...]
shell> tools/tpm install
```

The above configuration (and the default) will assume that your database has been configured with SSL enabled. The installation will error and fail if this is not the case. You must manually ensure database SSL has been enabled prior to issuing the install. Steps to enable this can be found in Section 6.10.1, "Enabling Database SSL"

If you DO NOT want to enable database level SSL, then you must also include the following option in the tpm configure command above:

```
--enable-connector-ssl=false
--datasource-enable-ssl=false
```

### Important

Installing from a staging host will automatically generate certificates and configuration for a secured installation. No further changes or actions are required.

Enabling Post-Installation

If, at install time, you disabled security (by specifying `--disable-security-controls=true` [404]) you can enable it by changing the value to false.

```
shell> tools/tpm configure defaults --disable-security-controls=false
shell> tools/tpm update --replace-jgroups-certificate --replace-tls-certificate --replace-release
```

The above configuration will assume that your database has been configured with SSL enabled. The update will error and fail if this is not the case. You must manually ensure database SSL has been enabled prior to issuing the update. Steps to enable this can be found in Section 6.10.1, "Enabling Database SSL"

If you DO NOT want to enable database level SSL, then you must also include the following options in the tpm configure command above:

```
--enable-connector-ssl=false
--datasource-enable-ssl=false
```

Following the update, you will also need to manually re-sync the certificates and keystores to all other nodes within your configuration. The following example uses scp for the copy and uses db1 as the primary source for the files to be copied. Adjust accordingly for your environment.

1. Sync Certificates and Keystores to all nodes

```
db1> for host in db2 db3 db4 db5 db6; do
scp /opt/continuent/share/[jpt]* ${host}:/opt/continuent/share
scp /opt/continuent/share/.[jpt]* ${host}:/opt/continuent/share
done
```

2. Restart all components, on all hosts

```
shell> replicator restart
```

### Warning

This update will force replicator processes to be restarted.

## 6.1.2. Enabling Security using the INI Method

Security will be enabled during initial install by default, should you choose to disable at install, then these steps will guide you in the process to enable as part of a post-install update

Enabled During Install

As mentioned, security is enabled by default. This is controlled by the `disable-security-controls` [404] property. If not supplied, the default is false. You can choose to specify this in your configuration for transparency if you wish.

```
disable-security-controls=false [404]
```

The above configuration (and the default) will assume that your database has been configured with SSL enabled. The installation will error and fail if this is not the case. You must manually ensure database SSL has been enabled prior to issuing the install. Steps to enable this can be found in Section 6.10.1, "Enabling Database SSL"

If you DO NOT want to enable database level SSL, then you must also include the following options in your `tungsten.ini` file:

```
datasource-enable-ssl=false [400]
```

Following installation there are a few additional steps that will be required before starting the software.

- You must select one of the nodes and copy that node's certificate/keystore/truststore files to all other nodes.

  Available as of Version 7.1.0, the tpm copy command can perform the file transfers for you. For example, run it from node db1 to copy to all the rest of the nodes in the cluster:

```
shell> tpm copy
About to copy all needed files for:
>>> Security directory: /opt/continuent/share
```

```
Please confirm that all nodes are done installing, and that none of the Tungsten processes have been started yet.
Ready to proceed (y/N)? y
```

For example, assuming you choose db1, and have 5 other nodes to copy the files to you could use this syntax:

```
shell> for host in db2 db3 db4 db5 db6; do
scp /opt/continuent/share/[jpt]* ${host}:/opt/continuent/share/
scp /opt/continuent/share/.[jpt]* ${host}:/opt/continuent/share/
done
```

### Important

The above example assumes ssh has been setup between nodes as the tungsten OS user. If this is not the case you will need to use whichever methods you have available to sync these files.

- Then, on all nodes, you can start the software:

```
shell> source /opt/continuent/share/env.sh
shell> startall
```

Enabling Post-Installation

If, at install time, you disabled security (by specifying `disable-security-controls=true` [404] you can enable it by changing the value to false in your `tungsten.ini` on all nodes.

The above configuration (and the default) will assume that your database has been configured with SSL enabled. The update will error and fail if this is not the case. You must manually ensure database SSL has been enabled prior to issuing the update. Steps to enable this can be found in Section 6.10.1, "Enabling Database SSL"

If you DO NOT want to enable database level SSL, then you must also include the following options in your `tungsten.ini` file:

```
datasource-enable-ssl=false [400]
```

Before issuing the update, there are a number of additional steps required. These are outlined below:

- First, configure the `tungsten.ini` file as follows:

```
disable-security-controls=false [404]
start-and-report=false [424]
```

- Do the update on each node, which will generate new, different certificates on every node.

### Warning

This update procedure will force replicators to be restarted.

```
shell> stopall
shell> tpm query staging
shell> cd {staging_directory}
shell> tools/tpm update --replace-jgroups-certificate --replace-tls-certificate --replace-release
```

- As with a fresh install, you must then select one of the nodes and copy that node's certificate files to all other nodes:

  Available as of Version 7.1.0, the tpm copy command can perform the file transfers for you. For example, run it from node db1 to copy to all the rest of the nodes:

```
shell> tpm copy
About to copy all needed files for:
>>> Security directory: /opt/continuent/share
Please confirm that all nodes are done installing, and that none of the Tungsten processes have been started yet.
Ready to proceed (y/N)? y
```

  For example, assuming you choose db1, and have 5 other nodes to copy the files to you could use this syntax:

```
shell> for host in db2 db3 db4 db5 db6; do
scp /opt/continuent/share/[jpt]* ${host}:/opt/continuent/share/
scp /opt/continuent/share/.[jpt]* ${host}:/opt/continuent/share/
done
```

### Important

The above example assumes ssh has been setup between nodes as the tungsten OS user. If this is not the case you will need to use whichever methods you have available to sync these files.

- On all nodes:

```
shell> startall
```

# 6.2. Disabling Security

There may be situations where you wish to disable securityfor the entire installation.

Security can be disabled in the following ways during configuration with tpm:

```
--disable-security-controls=true [404]
```

Disabling security through this single option, has the same effect as adding:

- `--file-protection-level=none` [408]

  Disables file level protection, including ownership and file mode settings.

- `--rmi-ssl=false` [407]

  Disables the use of SSL/TLS for communicating with services, this includes starting, stopping, or controlling individual services and operations, such as putting Tungsten Replicator online or offline.

- `--thl-ssl=false` [408]

  Disables the use of SSL/TLS for THL transmission between replicators.

- `--rmi-authentication=false` [407]

  Disables the use of authentication when accessing and controlling services.

- `--datasource-enable-ssl=false` [400]

- `--replicator-rest-api-ssl=false` [421]

  Disables SSL for communication with the Replicator API. This does not disable the API altogether. To do that, refer to `replicator-rest-api` [421]

# 6.3. Creating Suitable Certificates

By default, tpm can automatically create suitable certificates and configuration for use in your deployment. To create the required certificates by hand, use one of the following procedures.

## 6.3.1. Creating Tungsten Internal Certificates Using tpm cert

Available as of Version 7.1.0, the tpm cert command will perform the generation steps for you.

- Generating a TLS Certificate

  Run this command to create the TLS keystore `tungsten_tls_keystore.jks` in `$CONTINUENT_ROOT/generated`. You may use your own location, please see Section 9.5.2.8, “tpm cert: Getting Started - Advanced Example” for the steps required to do so.

```
## Perform a dry run generation of the file
shell> tpm cert gen tls_keystore --dryrun

## Perform a dry run generation of the file, using the shorter syntax, same as above
shell> tpm cert gen tls -n

## Generate the file, displaying the command executed with -x
shell> tpm cert gen tls -x
```

## 6.3.2. Creating Tungsten Internal Certificates Manually

To manually generate the security files, use the steps below:

- Generating a TLS Certificate

  Run this command to create the keystore in `/etc/tungsten/secure`. You may use your own location, but the values for `-storepass` and `-keypass` must match.

```
shell> keytool -genkey -alias tls \
-validity 3650 \
-keyalg RSA -keystore /etc/tungsten/secure/tungsten_tls_keystore.jks \
```

```
-dname "cn=Continuent, ou=IT, o=Continuent, c=US" \
-storepass mykeystorepass -keypass mykeystorepass
```

# 6.4. Installing from a Staging Host with Custom Certificates

Follow the steps in Section 6.3, “Creating Suitable Certificates” to create the TLS certificate.

Update your configuration to specify the certificate and the keystore password:

## 6.4.1. Installing from a Staging Host with Manually-Generated Certificates

```
shell> tools/tpm configure SERVICE \
    --java-tls-keystore-path=/etc/tungsten/secure/tungsten_tls_keystore.jks \
    --java-keystore-password=mykeystorepass
```

## 6.4.2. Installing from a Staging Host with Certificates Generated by tpm cert

Available as of Version 7.1.0, the tpm cert command can perform the generation steps for you. If you used tpm cert to generate files for you, they will be located in the `$CONTINUENT_ROOT/generated` directory by default.

```
shell> tools/tpm configure SERVICE \
    --java-tls-keystore-path=/opt/continuent/generated/tungsten_tls_keystore.jks \
    --java-keystore-password=mykeystorepass
```

# 6.5. Installing via INI File with Custom Certificates

Follow the steps in Section 6.3, “Creating Suitable Certificates” to create the TLS certificate.

## 6.5.1. Installing via INI File with Manually-Generated Certificates

1. Transfer the generated certificates to the same path on all hosts.
2. Update your configuration to specify the certificate and the keystore password:

   ```
   java-tls-keystore-path=/etc/tungsten/secure/tungsten_tls_keystore.jks
   java-keystore-password=mykeystorepass
   ```

## 6.5.2. Installing via INI File with Certificates Generated by tpm cert

1. Transfer the generated certificates to the same path on all hosts using your preferred method.

   Available as of Version 7.1.0, the tpm copy command can copy the generated files to all hosts for you if you have password-less SSH configured to all nodes.

   ```
   ## Perform a dry-run pass (-n) to test SSH
   ## and display the commands that would have been run
   ## to copy the generated files
   shell> tpm copy --gen -n

   ## Copy the generated files
   ## and display the command executed (-x)
   shell> tpm copy --gen -x
   ```

2. Update your configuration to specify the certificate and the keystore password:

   If you used the tpm cert command (available as of v7.1.0) to generate files for you, they will be located in the `$CONTINUENT_ROOT/generated` directory by default.

   ```
   java-tls-keystore-path=/opt/continuent/generated/tungsten_tls_keystore.jks
   java-keystore-password=mykeystorepass
   ```

# 6.6. Installing via INI File with CA-Signed Certificates

- This procedure will take a signed certificate from a known Certificate Authority and use it as the basis for all SSL operations within the replicator.
- The below example procedure assumes that you have an existing, installed and running Primary/Replica topology with security enabled by setting `disable-security-controls=false` [404]

  Assume a simple topology with with member hosts `db1` and `db2`

### Warning

> In all examples below, because you are updating an existing secure installation, the password `tungsten` is required, do not change it.

- Select one node to create the proper set of certs, i.e. `db1`:

```
shell> su - tungsten
shell> mkdir /etc/tungsten/secure
shell> mkdir ~/certs
shell> cd ~/certs
```

- Copy the available files (CA cert, Intermediate cert (if needed), signed cert and signing key) into ~/certs/, i.e.:

```
ca.crt.pem
int.crt.pem
signed.crt.pem
signing.key.pem
```

- Create a pkcs12 (.p12) version of the signed certificate:

```
shell> openssl pkcs12 -export -in ~/certs/signed.crt.pem -inkey ~/certs/signing.key.pem \
  -out ~/certs/tungsten_sec.crt.p12 -name replserver
Enter Export Password: tungsten
Verifying - Enter Export Password: tungsten
```

### Important

> When using OpenSSL 3.0 with Java 1.8, you MUST add the `-legacy` option to the openssl command.

- Create a pkcs12-based keystore (.jks) version of the signed certificate:

```
shell> keytool -importkeystore -deststorepass tungsten -destkeystore /etc/tungsten/secure/tungsten_keystore.jks \
  -srckeystore ~/certs/tungsten_sec.crt.p12 -srcstoretype pkcs12 -deststoretype pkcs12
Importing keystore /home/tungsten/certs/tungsten_sec.crt.p12 to /etc/tungsten/secure/tungsten_keystore.jks...
Enter source keystore password:  tungsten
Entry for alias replserver successfully imported.
Import command completed:  1 entries successfully imported, 0 entries failed or cancelled
```

- Import the Certificate Authority's certificate into the keystore:

```
shell> keytool -import -alias careplserver -file ~/certs/ca.crt.pem -keypass tungsten \
  -keystore /etc/tungsten/secure/tungsten_keystore.jks -storepass tungsten
...
Trust this certificate? [no]:  yes
Certificate was added to keystore
```

- Import the Certificate Authority's intermediate certificate (if supplied) into the keystore:

```
shell> keytool -import -alias interreplserver -file ~/certs/int.crt.pem -keypass tungsten \
  -keystore /etc/tungsten/secure/tungsten_keystore.jks -storepass tungsten
Certificate was added to keystore
```

- Export the cert from the keystore into file `client.cer` for use in the next step to create the truststore:

```
shell> keytool -export -alias replserver -file ~/certs/client.cer \
  -keystore /etc/tungsten/secure/tungsten_keystore.jks
Enter keystore password:  tungsten
Certificate stored in file </home/tungsten/certs/client.cer>
```

- Create the truststore:

```
shell> keytool -import -trustcacerts -alias replserver -file ~/certs/client.cer \
  -keystore /etc/tungsten/secure/tungsten_truststore.ts -storepass tungsten -noprompt
Certificate was added to keystore
```

- Create the rmi_jmx password store entry:

```
shell> tpasswd -c tungsten tungsten -t rmi_jmx -p /etc/tungsten/secure/passwords.store -e \
  -ts /etc/tungsten/secure/tungsten_truststore.ts -tsp tungsten
Using parameters:
-----------------
security.properties    = /opt/continuent/tungsten/cluster-home/../cluster-home/conf/security.properties
password.file.location    = /etc/tungsten/secure/passwords.store
encrypted.password    = true
truststore.location    = /etc/tungsten/secure/tungsten_truststore.ts
truststore.password    = *********
-----------------
Creating non existing file: /etc/tungsten/secure/passwords.store
```

```
User created successfuly: tungsten
```

- Create the tls password store entry:

```
shell> tpasswd -c tungsten tungsten -t unknown -p /etc/tungsten/secure/passwords.store -e \
  -ts /etc/tungsten/secure/tungsten_truststore.ts -tsp tungsten
Using parameters:
----------------
security.properties    = /opt/continuent/tungsten/cluster-home/../cluster-home/conf/security.properties
password.file.location    = /etc/tungsten/secure/passwords.store
encrypted.password    = true
truststore.location    = /etc/tungsten/secure/tungsten_truststore.ts
truststore.password    = ********
----------------
User created successfuly: tungsten
```

- List and verify the user for each security service password store entry, rmi_jmx and tls (which has a display tag of `unknown`):

```
shell> tpasswd -l -p /etc/tungsten/secure/passwords.store -ts /etc/tungsten/secure/tungsten_truststore.ts
Using parameters:
----------------
security.properties    = /opt/continuent/tungsten/cluster-home/../cluster-home/conf/security.properties
password.file.location    = ./passwords.store
encrypted.password    = true
truststore.location    = ./tungsten_truststore.ts
truststore.password    = ********
----------------
Listing users by application type:

[unknown]
-----------
tungsten

[rmi_jmx]
-----------
tungsten
```

- On host db1, transfer the generated certificates to the same path on all remaining hosts:

```
shell> for host in `seq 2 3`; do rsync -av /etc/tungsten/secure/ db$host:/etc/tungsten/secure/; done
```

- Edit the `/etc/tungsten/tungsten.ini` configuration file on all nodes and add:

```
[defaults]
...
disable-security-controls=false
java-keystore-path=/etc/tungsten/secure/tungsten_keystore.jks
java-keystore-password=tungsten
java-truststore-path=/etc/tungsten/secure/tungsten_truststore.ts
java-truststore-password=tungsten
rmi-ssl=true
rmi-authentication=true
rmi-user=tungsten
java-passwordstore-path=/etc/tungsten/secure/passwords.store
```

> **Important**
>
> When `java-keystore-path` [410] is passed to tpm, the keystore must contain both tls and mysql certs when appropriate. tpm will NOT add mysql cert nor generate tls cert when this flag is found, so both certs must be manually imported already.

- On ALL nodes, stop the replicator software, execute the update, then start the replicators:

> **Warning**
>
> This procedure requires the complete restart of all layers of the Cluster, and will cause a brief downtime.

```
shell> tpm query staging
shell> cd {staging_dir}
shell> stopall
shell> tools/tpm update --replace-release
shell> startall
```

## 6.7. Replacing the JGroups Certificate from a Staging Directory

If you meet the requirements to use an automatically generated certificate from the staging directory, the tpm update command can handle the certificate replacement. Simply add the `--replace-jgroups-certificate` option to your command. This will create errors if your staging configuration does not reflect the full list of hosts or if you limit the command to a specific host.

```
shell> tools/tpm update --replace-jgroups-certificate --replace-release
```

If you do not meet these requirements, generate a new certificate and update it through the tpm command.

```
shell> tools/tpm configure SERVICE \
      --java-jgroups-keystore-path=/etc/tungsten/jgroups.jceks \
      --java-keystore-password=mykeystorepass
```

Then perform an update and replace the entire release directory:

```
shell> tools/tpm update --replace-release
```

# 6.8. Replacing the TLS Certificate from a Staging Directory

If you meet the requirements to use an automatically generated certificate from the staging directory, the tpm update command can handle the certificate replacement. Simply add the `--replace-tls-certificate` [420] option to your command. This will create errors if your staging configuration does not reflect the full list of hosts or if you limit the command to a specific host.

```
shell> tools/tpm update --replace-tls-certificate --replace-release
```

If you do not meet these requirements, generate a new certificate and update it through the tpm command.

```
shell> tools/tpm configure SERVICE \
        --java-tls-keystore-path=/etc/tungsten/tls.jks \
        --java-keystore-password=mykeystorepass
```

Then perform an update and replace the entire release directory:

```
shell> tools/tpm update --replace-release
```

# 6.9. Removing TLS Encryption from a Staging Directory

Using the tpm update command, the general Continuent service encryption can be easily removed.

```
shell> tpm configure SERVICE \
   --thl-ssl=false \
   --rmi-ssl=false \
   --rmi-authentication=false
```

Then perform an update and replace the entire release directory:

```
shell> tpm update --replace-release
```

# 6.10. Enabling Tungsten<>Database Security

This section explains how to enable security between the database and various other parts of the topology, including:

- Database server SSL

  This is the first step, and the prerequiste for all the remaining steps. You must have the database server properly configured to support SSL before any of the other procedures will work.

  See Section 6.10.1, “Enabling Database SSL”

- Tungsten Replicator to the database server

  This usually happens during the second step, and what allows Tungsten Replicator to communicate securely with the database server.

  See Section 6.10.2, “Configure Tungsten<>Database Secure Communication”

## 6.10.1. Enabling Database SSL

The steps outlined below explain how to enable security within MySQL (If it is not already enabled by default in the release your are using). There are different approaches depending on the version/distribution of MySQL you are using. If in any doubt, you should consult the appropriate documentation pages for the MySQL release you are using.

### 6.10.1.1. Using the tpm cert gen mysqlcerts Command

Available as of Version 7.1.0, the tpm cert gen mysqlcerts command can perform the database certificate generation steps for you, along with handling directory creation, ownership and permissions.

You will need the following:

- The mysql_ssl_rsa_setup command must be available (shipped with MySQL 5.7 onwards)
- You will need to update the various configuration files as shown in Section 6.10.1.2, "Using mysql_ssl_rsa_setup utility"

```
shell> tpm cert gen my --datadir /etc/mysql/certs
About to execute Write Action `gen mysqlcerts`,
Ready to proceed (y/N)? y

======================================================================
>>> doGen processing typeSpec: mysqlcerts
======================================================================

gen::genMysqlCerts: Using datadir /etc/mysql/certs from the command line
gen::genMysqlCerts: datadir /etc/mysql/certs does not exist - attempting to create...SUCCESS
gen::genMysqlCerts: SUCCESS - Executed /usr/bin/sudo mysql_ssl_rsa_setup -d /etc/mysql/certs
----------------------------------------------------------------------
-rw-r----- 1 mysql mysql 1679 Aug 11 16:00 /etc/mysql/certs/ca-key.pem
-rw-r--r-- 1 mysql mysql 1107 Aug 11 16:00 /etc/mysql/certs/ca.pem
-rw-r--r-- 1 mysql mysql 1107 Aug 11 16:00 /etc/mysql/certs/client-cert.pem
-rw-r----- 1 mysql mysql 1679 Aug 11 16:00 /etc/mysql/certs/client-key.pem
-rw-r----- 1 mysql mysql 1675 Aug 11 16:00 /etc/mysql/certs/private_key.pem
-rw-r--r-- 1 mysql mysql 451 Aug 11 16:00 /etc/mysql/certs/public_key.pem
-rw-r--r-- 1 mysql mysql 1107 Aug 11 16:00 /etc/mysql/certs/server-cert.pem
-rw-r----- 1 mysql mysql 1679 Aug 11 16:00 /etc/mysql/certs/server-key.pem
```

### 6.10.1.2. Using mysql_ssl_rsa_setup utility

This tool is shipped with MySQL 5.7 onwards and makes the creation of all of the certificates much easier. If you have this tool available, then you can follow the steps below:

- Invoke mysql_ssl_rsa_setup on one of the hosts. This will generate the SSL certificates and RSA keys by default in `/var/lib/mysql`. These files should be copied to the other hosts.

  The mysql_ssl_rsa_setup supports the `--datadir=/my/custom/path/` option if the you want to use a different location. Continuent recommends using `/etc/mysql/certs` as the location.

  **Note**

  The generated pem files should be readable by the `tungsten` and `mysql` OS users.

- Add the following to the `[mysqld]` stanza in your `my.cnf`

  ```
  [mysqld]
  ssl_ca=/etc/mysql/certs/ca.pem
  ssl_cert=/etc/mysql/certs/server-cert.pem
  ssl_key=/etc/mysql/certs/server-key.pem
  require_secure_transport=ON
  ```

- Add the following to the `>[client]` stanz in your `my.cnf`

  ```
  [client]
  ssl_ca=/etc/mysql/certs/ca.pem
  ssl_cert=/etc/mysql/certs/client-cert.pem
  ssl_key=/etc/mysql/certs/client-key.pem
  ssl_mode=REQUIRED
  ```

  This will allow the mysql client will connect through SSL to the server.

- tpm will parse the `my.cnf` file and retrieve the certificates paths. It is still possible to specify different paths via the following `tungsten.ini` settings:

  ```
  repl-datasource-mysql-ssl-ca=/etc/mysql/certs/ca.pem
  repl-datasource-mysql-ssl-cert=/etc/mysql/certs/client-cert.pem
  repl-datasource-mysql-ssl-key=/etc/mysql/certs/client-key.pem
  ```

  tpm install will add these client certificates to the tungsten truststore, keystore.

### 6.10.1.3. Manually Creating Certificates

**Important**

The "Common Name" field for the Server and Client certificates MUST be different than the "Common Name" specified for the CA Cert.

1. Generate CA Cert

```
shell> openssl genrsa 2048 > $MYSQL_CERTS_PATH/ca-key.pem

shell> openssl req -sha256 -new -x509 -nodes -days 3650 \
-key $MYSQL_CERTS_PATH/ca-key.pem \
-out $MYSQL_CERTS_PATH/ca.pem
```

2. Generate Server Cert

```
shell> openssl req -sha256 -newkey rsa:2048 -nodes -days 3650 \
-keyout $MYSQL_CERTS_PATH/server-key.pem \
-out $MYSQL_CERTS_PATH/server-req.pem

shell> openssl rsa -in $MYSQL_CERTS_PATH/server-key.pem -out $MYSQL_CERTS_PATH/server-key.pem

shell> openssl x509 -sha256 -req -in $MYSQL_CERTS_PATH/server-req.pem -days 3650 \
-CA $MYSQL_CERTS_PATH/ca.pem \
-CAkey $MYSQL_CERTS_PATH/ca-key.pem \
-set_serial 01 \
-out $MYSQL_CERTS_PATH/server-cert.pem
```

3. Generate Client Cert

```
shell> openssl req -sha256 -newkey rsa:2048 -days 3600 -nodes \
-keyout $MYSQL_CERTS_PATH/client-key.pem \
-out $MYSQL_CERTS_PATH/client-req.pem

shell> openssl rsa -in $MYSQL_CERTS_PATH/client-key.pem -out $MYSQL_CERTS_PATH/client-key.pem

shell> openssl x509 -sha256 -req -in $MYSQL_CERTS_PATH/client-req.pem -days 3650 \
-CA $MYSQL_CERTS_PATH/ca.pem \
-CAkey $MYSQL_CERTS_PATH/ca-key.pem \
-set_serial 01 \
-out $MYSQL_CERTS_PATH/client-cert.pem
```

4. Verify All Certificates

```
shell> openssl verify -CAfile $MYSQL_CERTS_PATH/ca.pem \
$MYSQL_CERTS_PATH/server-cert.pem $MYSQL_CERTS_PATH/client-cert.pem
```

5. Copy certs to all Database nodes (repeat as needed so that every Database node has the same certificates)

```
shell> rsync -av $MYSQL_CERTS_PATH/ yourDBhost:$MYSQL_CERTS_PATH/
```

6. Set proper ownership and permissions on ALL DB nodes

```
shell> sudo chown -R mysql: $MYSQL_CERTS_PATH/
shell> sudo chmod -R g+w $MYSQL_CERTS_PATH/
```

7. Update the `my.cnf` file to include the SSL certificates you just created (add three lines to the `[mysqld]` stanza)

```
shell> vi /etc/my.cnf
[mysqld]
...
port=13306
# add three lines for SSL support
ssl-ca=/etc/mysql/certs/ca.pem
ssl-cert=/etc/mysql/certs/server-cert.pem
ssl-key=/etc/mysql/certs/server-key.pem
...
```

8. Restart MySQL on all nodes using the standard rolling maintenance procedure - see Section 7.13.3, “Performing Maintenance on an Entire Dataservice” for more information.

```
cctrl> ls
cctrl> datasource db3 shun
db3# service mysql restart
cctrl> recover

cctrl> datasource db2 shun
db2# service mysql restart
cctrl> recover

cctrl> switch to db2

cctrl> datasource db1 shun
db1# service mysql restart
cctrl> recover

cctrl> switch to db1
```

```
cctrl> ls
```

9. Add a new user to MySQL that requires SSL to connect. Do this just once on the current Primary and let it propagate to the Replicas.

```
shell> tpm mysql
mysql> DROP USER ssl_user;
mysql> CREATE USER ssl_user@'%' IDENTIFIED BY 'secret';
mysql> GRANT ALL ON *.* TO ssl_user@'%' REQUIRE SSL WITH GRANT OPTION;
mysql> flush privileges;
```

10. Verify that MySQL is working with SSL

    a. Expect this to fail, because the ssl_user is only allowed to connect to the database using SSL:

    ```
    shell> mysql -u ssl_user -psecret -h 127.0.0.1 -P 13306
    ```

    b. Expect this to pass, because we have supplied the proper SSL credentials:

    ```
    shell> mysql -u ssl_user -psecret -h 127.0.0.1 -P 13306 --ssl-ca=/etc/mysql/certs/ca.pem
    ```

    c. Verify SSL:

    ```
    mysql> status
    ...
    SSL:  Cipher in use is DHE-RSA-AES256-SHA
    ...
    ```

    **Important**

    If you are able to login to MySQL and see that the status is SSL: Cipher in use, then you have successfully configured MySQL to use SSL.

### 6.10.1.4. Enabling Database Level SSL with Amazon AWS Aurora

To enable Tungsten Replicator to communicate with Amazon Aurora, via SSL, the following simple steps can be followed.

- Obtain the certificate from Amazon appropriate for the region that your Aurora instance in hosted. More information can be found here.
- Copy the file to the Tungsten Replicator host into a directory of your choice.
- Add the following properties to your configuration. (In this example our certificate is within `/opt/continuent/share`. Adjust to suit your environment)

```
property=replicator.datasource.global.connectionSpec.urlOptions=noPrepStmtCache=true&
» serverCertificate=/opt/continuent/share/rds-ca-2019-eu-west-1.pem
datasource-enable-ssl=true
```

- You can now install, or if the replicator was already installed, issue an update

## 6.10.2. Configure Tungsten<>Database Secure Communication

If you choose to enable database level SSL within your MySQL installation, there are a number of additional steps required to allow the Tungsten Components to be able to communicate to the database layer.

The steps below make the following assumptions:

- You have enabled SSL using the correct procedures for your distribution of MySQL. If not, refer to Section 6.10.1, “Enabling Database SSL”.
- You have generated, and have access to, the client level certificates and keys
- If you are installing an Offboard extractor/applier, the client certificates and keys have been copied to the extractor/applier hosts

1. If SSL has been enabled within the Tungsten installation, then you should either have the following parameter in your configuration, or it will be ommitted altogether since security is enabled by default:

   ```
   disable-security-controls=false
   ```

   As a result, you should have a number of files within `/opt/continuent/share`

   ```
   shell> ls -l
   total 20
   -rw-rw-r-- 1 tungsten tungsten  104 Jul 18 10:15 jmxremote.access
   -rw-rw-r-- 1 tungsten tungsten  729 Jul 18 10:15 passwords.store
   -rw-rw-r-- 1 tungsten tungsten 2268 Jul 18 10:15 tungsten_keystore.jks
   -rw-rw-r-- 1 tungsten tungsten 1079 Jul 18 10:15 tungsten_truststore.ts
   ```

2. If you do not have SSL enabled within the installation and you require this, then follow the steps in Section 6.1, "Enabling Security" first

3. If you do not require SSL between the Replicators, and only require SSL between the replicator and the database, then add the following parameters to your configuration, but do not run tpm update yet.

```
java-truststore-path=/home/tungsten/tungsten_truststore.ts
java-truststore-password=tungsten
java-keystore-path=/home/tungsten/tungsten_keystore.jks
```

4. Next, add the following parameters to your installation, but do not run tpm update yet:

```
datasource-enable-ssl=true
```

5. You now need to convert the mysql client key to PKCS12 format. Adjust the path and filename in the example to suit your environment

```
shell> openssl pkcs12 -export -in /home/tungsten/client-cert.pem \
-inkey /home/tungsten/client-key.pem \
-name mysql -out /home/tungsten/client-key.p12
```

> **Important**
>
> When prompted for a password, you MUST enter tungsten

> **Important**
>
> When using OpenSSL 3.0 with Java 1.8, you MUST add the `-legacy` option to the openssl command.

6. You now need to import the key, either into the existing keystore if it exists, or into a new one if SSL is not being enabled at the replicator level

   If Tungsten level SSL has been enabled

```
shell> keytool -importkeystore -deststorepass tungsten \
-destkeystore /opt/continuent/share/tungsten_keystore.jks \
-srckeystore /home/tungsten/client-key.p12 -srcstoretype PKCS12
```

   If ONLY Database SSL is required

```
shell> keytool -importkeystore -deststorepass tungsten \
-destkeystore /home/tungsten/tungsten_keystore.jks \
-srckeystore /home/tungsten/client-key.p12 -srcstoretype PKCS12
```

   When prompted for a password, enter tungsten

7. Next, import the client certificate into the truststore

   If Tungsten level SSL has been enabled

```
shell> keytool -import -alias mysql -trustcacerts -file /home/tungsten/ca.pem \
-keystore /opt/continuent/share/tungsten_truststore.ts
```

   If ONLY Database SSL is required

```
shell> keytool -import -alias mysql -trustcacerts -file /home/tungsten/ca.pem \
-keystore /home/tungsten/tungsten_truststore.ts
```

   When prompted for a password, enter tungsten

8. Finally, and only if Tungsten level SSL has been enabled, we need to create backups copies of the keystore and truststore as follows:

```
shell> cp /opt/continuent/share/tungsten_truststore.ts /opt/continuent/share/.tungsten_truststore.ts.orig
shell> cp /opt/continuent/share/tungsten_keystore.jks /opt/continuent/share/.tungsten_keystore.jks.orig
```

9. Issue tpm update to apply the configuration

The replicators will be restarted as part of the update process, and should now be using SSL to connect successfully to MySQL

# Chapter 7. Operations Guide

There are a number of key operations that enable you to monitor and manage your replication cluster. Tungsten Replicator includes a small number of tools that can help with this process, including the core trepctl command, for controlling the replication system, and thl, which provides an interface to the Tungsten History Log and information about the changes that have been recorded to the log and distributed to the Targets.

During the installation process the file `/opt/continuent/share/env.sh` will have been created which will seed the shell with the necessary `$PATH` and other details to more easily manage your cluster. You can load this script manually using:

```
shell> source /opt/continuent/share/env.sh
```

Once loaded, all of the tools for controlling and monitoring your replicator installation should be part of your standard `PATH`.

## 7.1. The Home Directory

After installing Tungsten Replicator the home directory will be filled with a set of new directories. The home directory is specified by `--home-directory` [409] or `--install-directory` [409]. If you have multiple installations on a single server; each directory will include the same entries.

- tungsten - A symlink to the most recent version of the software. The symlink points into the `releases` directory. You should always use the symlink to ensure the most recent configuration and software is used.
- releases - Storage for the current and previous versions of the software. During an upgrade the new software will be copied into this directory and the `tungsten` symlink will be updated. See Section D.1.2, "The `releases` Directory" for more information.
- service_logs - Includes symlinks to the primary log for the replicator, manager and connector. This directory also includes logs for other tools distributed for Tungsten Cluster.
- backups - Storage for backup files created through trepctl. See Section D.1.1, "The `backups` Directory" for more information.
- thl - Storage for THL files created by the replicator. Each replication service gets a dedicated sub-directory for storing THL files. See Section D.1.5, "The `thl` Directory" for more information.
- relay - Temporary storage for downloaded MySQL binary logs before they are converted into THL files.
- share - Storage for files that must persist between different software versions. The `env.sh` script will setup your shell environment to allow easy access to Tungsten Cluster tools.

## 7.2. Establishing the Shell Environment

The tools required to operate Tungsten Cluster are located in many directories around the home directory. The best way to access them is by setting up your shell environment.

The `env.sh` file will automatically be included if you specify the `--profile-script` [417] during installation. This option may be included during a configuration change with tpm update.

If the `env.sh` file hasn't been included you may do so by hand with source.

```
shell> source /opt/continuent/share/env.sh
```

> **Important**
>
> Special consideration must be taken if you have multiple installations on a single server. That applies for clustering and replication or multiple replicators.
>
> Include the `--executable-prefix` [408] and `--profile-script` [417] options in your configuration. Instead of extending the `$PATH` variable; the `env.sh` script will define aliases for each command. If you specified `--executable-prefix=mm` [408] the trepctl command would be accessed as mm_trepctl.

## 7.3. Understanding Replicator Roles

Replicators can have one of five roles, Extractor(master),Applier(slave), thl-server, thl-client or thl-applier.

- *master*

A replicator in a *master* role extracts data from a source database (for example, by reading the binary log from a MySQL server), and generates THL. As a *master* the replicator also provides the THL to other replicators over the network connection.

- *slave*

  A *slave* replicator pulls THL data from a *master* and then applies that data to a target database.

- *thl-server* [192]

  A *thl-server* [192] replicator is a special role that Extractor replicators can be changed to temporarily when a Primary is taken offline. This will allow downstream Applier replicators to download and apply any THL that hasn't yet been processed by the Applier.

  To enable this role, issue the following statements

  ```
  shell> trepctl offline
  shell> trepctl setrole -role thl-server
  shell> trepctl online
  ```

  To revert back to the original Extractor role, issue the following

  ```
  shell> trepctl offline
  shell> trepctl setrole -role master
  shell> trepctl online
  ```

- *thl-client* [192]

  A *thl-client* [192] replicator is a special role that Applier replicators can be changed to. This will allow the Applier replicator to download any THL available from the upstream Extractor, but does NOT apply the THL to the target database.

  To enable this role, issue the following statements

  ```
  shell> trepctl offline
  shell> trepctl setrole -role thl-client
  shell> trepctl online
  ```

  To revert back to the original Applier role, issue the following

  ```
  shell> trepctl offline
  shell> trepctl setrole -role slave
  shell> trepctl online
  ```

- *thl-applier* [192]

  A *thl-applier* [192] replicator is a special role that applier replicators can be changed to temporarily when a Primary is taken offline. This will allow downstream applier replicators to apply any locally available THL that hasn't yet been processed by the applier.

  To enable this role, issue the following statements

  ```
  shell> trepctl offline
  shell> trepctl setrole -role thl-applier
  shell> trepctl online
  ```

  To revert back to the original role, issue the following

  ```
  shell> trepctl offline
  shell> trepctl setrole -role slave
  shell> trepctl online
  ```

# 7.4. Checking Replication Status

To check the replication status you can use the trepctl command. This accepts a number of command-specific verbs that provide status and control information for your configured cluster. The basic format of the command is:

```
shell> trepctl [-host hostname] command
```

The `-host` option is not required, and enables you to check the status of a different host than the current node.

To get the basic information about the currently configured services on a node and current status, use the services verb command:

```
shell> trepctl services
Processing services command...
NAME              VALUE
----              -----
appliedLastSeqno: 211
```

```
appliedLatency  : 17.66
role            : slave
serviceName     : firstrep
serviceType     : local
started         : true
state           : ONLINE
Finished services command...
```

In the above example, the output shows the last sequence number and latency of the host, in this case an Applier, compared to the Extractor from which it is processing information. In this example, the last sequence number and the latency between that sequence being processed on the Extractor and applied to the Target is 17.66 seconds. You can compare this information to that provided by the Extractor, either by logging into the Extractor and running the same command, or by using the host command-line option:

```
shell> trepctl -host host1 services
Processing services command...
NAME              VALUE
----              -----
appliedLastSeqno: 365
appliedLatency  : 0.614
role            : master
serviceName     : firstrep
serviceType     : local
started         : true
state           : ONLINE
Finished services command...
```

By comparing the *appliedLastSeqno* for the Extractor against the value on the Applier, it is possible to determine that the Applier and the Extractor are not yet synchronized.

For a more detailed output of the current status, use the status command, which provides much more detailed output of the current replication status:

```
shell> trepctl status
Processing status command...
NAME                     VALUE
----                     -----
appliedLastEventId     : mysql-bin.000064:0000000002757461;0
appliedLastSeqno       : 212
appliedLatency         : 263.43
channels               : 1
clusterName            : default
currentEventId         : NONE
currentTimeMillis      : 1365082088916
dataServerHost         : host2
extensions             :
latestEpochNumber      : 0
masterConnectUri       : thl://host1:2112/
masterListenUri        : thl://host2:2112/
maximumStoredSeqNo     : 724
minimumStoredSeqNo     : 0
offlineRequests        : NONE
pendingError           : NONE
pendingErrorCode       : NONE
pendingErrorEventId    : NONE
pendingErrorSeqno      : -1
pendingExceptionMessage: NONE
pipelineSource         : thl://host1:2112/
relativeLatency        : 655.915
resourcePrecedence     : 99
rmiPort                : 10000
role                   : slave
seqnoType              : java.lang.Long
serviceName            : firstrep
serviceType            : local
simpleServiceName      : firstrep
siteName               : default
sourceId               : host2
state                  : ONLINE
timeInStateSeconds     : 893.32
uptimeSeconds          : 9370.031
version                : Tungsten Replicator 7.1.2 build 81
Finished status command...
```

Similar to the host specification, trepctl provides information for the default service. If you have installed multiple services, you must specify the service explicitly:

```
shell> trepctrl -service servicename status
```

If the service has been configured to operate on an alternative management port, this can be specified using the `-port` option. The default is to use port 10000.

The above command was executed on the Target host, `host2`. Some key parameter values from the generated output:

- *appliedLastEventId*

  This shows the last event from the source event stream that was applied to the database. In this case, the output shows that source of the data was a MySQL binary log. The portion before the colon, `mysql-bin.000064` is the filename of the binary log on the Source. The portion after the colon is the physical location, in bytes, within the binary log file.

- *appliedLastSeqno*

  The last sequence number for the transaction from the Tungsten stage that has been applied to the database. This indicates the last actual transaction information written into the Target database.

  When using parallel replication, this parameter returns the minimum applied sequence number among all the channels applying data.

- *appliedLatency*

  The *appliedLatency* is the latency between the commit time and the time the last committed transaction reached the end of the corresponding pipeline within the replicator.

  In replicators that are operating with parallel apply, *appliedLatency* indicates the latency of the trailing channel. Because the parallel apply mechanism does not update all channels simultaneously, the figure shown may trail significantly from the actual latency.

- *masterConnectUri*

  On an Extractor, the value will be empty.

  On an Applier, the URI of the Extractor Tungsten Replicator from which the transaction data is being read from. The value supports multiple URIs (separated by comma) for topologies with multiple Sources.

- *maximumStoredSeqNo*

  The maximum transaction ID that has been stored locally on the machine in the THL. Because Tungsten Replicator operates in stages, it is sometimes important to compare the sequence and latency between information being ready from the source into the THL, and then from the THL into the database. You can compare this value to the *appliedLastSeqno*, which indicates the last sequence committed to the database. The information is provided at a resolution of milliseconds.

- *pipelineSource*

  Indicates the source of the information that is written into the THL. For an Extractor, *pipelineSource* is the MySQL binary log. For an Applier, *pipelineSource* is the THL of the Extractor.

- *relativeLatency*

  The relativeLatency is the latency between now and timestamp of the last event written into the local THL. An increasing *relativeLatency* indicates that the replicator may have stalled and stopped applying changes to the dataserver.

- *state*

  Shows the current status for this node. In the event of a failure, the status will indicate that the node is in a state other than `ONLINE` [195]. The *timeInStateSeconds* will indicate how long the node has been in that state, and therefore how long the node may have been down or unavailable.

The easiest method to check the health of your replication is to compare the current sequence numbers and latencies for each Applier compared to the Extractor. For example:

```
shell> trepctl -host host2 status|grep applied
appliedLastEventId     : mysql-bin.000076:0000000087725114;0
appliedLastSeqno       : 2445
appliedLatency         : 252.0
...
shell> trepctl -host host1 status|grep applied
appliedLastEventId     : mysql-bin.000076:0000000087725114;0
appliedLastSeqno       : 2445
appliedLatency         : 2.515
```

### Note

For parallel replication and complex multi-service replication structures, there are additional parameters and information to consider when checking and confirming the health of the cluster.

The above indicates that the two hosts are up to date, but that there is a significant latency on the Applier for performing updates.

> Tungsten Replicator Schema
>
> Tungsten Replicator creates and updates information in a special schema created within the database which contains more specific information about the replication information transferred. The schema is named according to the servicename of the replication configuration, for example if the server is `firstrep`, the schema will be `tungsten_firstrep`.
>
> The sequence number of the last transferred and applied transaction is recorded in the `trep_commit_seqno` table.

## 7.4.1. Understanding Replicator States

Each node within the cluster will have a specific state that indicates whether the node is up and running and servicing requests, or whether there is a fault or problem. Understanding these states will enable you to clearly identify the current operational status of your nodes and cluster as a whole.

A list of the possible states for the replicator includes:

- `START` [195]

  The replicator service is starting up and reading the replicator properties configuration file.

- `OFFLINE:NORMAL` [195]

  The node has been deliberately placed into the offline mode by an administrator. No replication events are processed, and reading or writing to the underlying database does not take place.

- `OFFLINE:ERROR` [195]

  The node has entered the offline state because of an error. No replication events are processed, and reading or writing to the underlying database does not take place.

- `GOING-ONLINE:PROVISIONING` [195]

  The replicator is currently reading provisioning information from the Source database before entering the `ONLINE` [195] state.

- `GOING-ONLINE:RESTORING` [195]

  The replicator is preparing to go online and is currently restoring data from a backup.

- `GOING-ONLINE:SYNCHRONIZING` [195]

  The replicator is preparing to go online and is currently preparing to process any outstanding events from the incoming event stream. This mode occurs when an Applier has been switched online after maintenance, or in the event of a temporary network error where the Applier has reconnected to the Extractor.

- `ONLINE` [195]

  The node is currently online and processing events, reading incoming data and applying those changes to the database as required. In this mode the current status and position within the replication stream is recorded and can be monitored. Replication will continue until an error or administrative condition switches the node into the `OFFLINE` [195] state.

- `GOING-OFFLINE` [195]

  The replicator is processing any outstanding events or transactions that were in progress when the node was switched offline. When these transactions are complete, and the resources in use (memory, network connections) have been closed down, the replicator will switch to the `OFFLINE:NORMAL` [195] state. This state may also be seen in a node where auto-enable is disabled after a start or restart operation.

- `ONLINE:DEGRADED` [195]

  This status will be seen on an `Extractor` replicator and is indicative of the replicator loosing connectivity to the Source Database that it is extracting from. The replicator will still continue to extract entries from the binary log that have not yet been processed. After extracting all log entries, the replicator will proceed to the `ONLINE:DEGRADED-BINLOG-FULLY-READ` [195] state.

- `ONLINE:DEGRADED-BINLOG-FULLY-READ` [195]

  This status will be seen on an `Extractor` replicator following the `ONLINE:DEGRADED` [195] state and indicates that the replicator has completed reading all binglog entries. In a clustering environment, it indicates to the cluster that failover can now proceed.

In general, the state of a node during operation will go through a natural progression within certain situations. In normal operation, assuming no failures or problems, and no management requested offline, a node will remain in the `ONLINE` [195] state indefinitely.

Maintenance on Tungsten Replicator or the dataserver must be performed while in the OFFLINE [195] state. In the OFFLINE [195] state, write locks on the THL and other files are released, and reads or writes from the dataserver are stopped until the replicator is ONLINE [195] again.

## 7.4.2. Replicator States During Operations

During a maintenance operation, a node will typically go through the following states at different points of the operation:

| Operation | State |
|---|---|
| Node operating normally | ONLINE [195] |
| Administrator puts node into offline state | GOING-OFFLINE [195] |
| Node is offline | OFFLINE:NORMAL [195] |
| Administrator puts node into online state | GOING-ONLINE:SYNCHRONIZING [195] |
| Node catches up with Extractor | ONLINE [195] |

In the event of a failure, the sequence will trigger the node into the error state and then recovery into the online state:

| Operation | State |
|---|---|
| Node operating normally | ONLINE [195] |
| Failure causes the node to go offline | OFFLINE:ERROR [195] |
| Administrator fixes error and puts node into online state | GOING-ONLINE:SYNCHRONIZING [195] |
| Node catches up with Extractor | ONLINE [195] |

During an error state where a backup of the data is restored to a node in preparation of bringing the node back into operation:

| Operation | State |
|---|---|
| Node operating normally | ONLINE [195] |
| Failure causes the node to go offline | OFFLINE:ERROR [195] |
| Administrator restores node from backup data | GOING-ONLINE:RESTORING [195] |
| Once restore is complete, node synchronizes with the Extractor | GOING-ONLINE:SYNCHRONIZING [195] |
| Node catches up with Extractor | ONLINE [195] |

## 7.4.3. Changing Replicator States

You can manually change the replicator states on any node by using the trepctl command.

To switch to the OFFLINE [195] state if you are currently ONLINE [195]:

```
shell> trepctl offline
```

Unless there is an error, no information is reported. The current state can be verified using the trepctl status:

```
shell> trepctl status
Processing status command...
...
state                    : OFFLINE:NORMAL
timeInStateSeconds       : 21.409
uptimeSeconds            : 935.072
```

To switch back to the ONLINE [195] state:

```
shell> trepctl online
```

When using replicator states in this manner, the replication between hosts is effectively paused. Any outstanding events from the Extractor will be replicated to the Applier with the replication continuing from the point where the node was switched to the OFFLINE [195] state. The sequence number and latency will be reported accordingly, as seen in the example below where the node is significantly behind the Primary:

```
shell> trepctl status
Processing status command...
NAME                      VALUE
----                      -----
appliedLastEventId       : mysql-bin.000004:0000000005162941;0
appliedLastSeqno         : 21
appliedLatency           : 179.366
```

# 7.5. Managing Transaction Failures

Inconsistencies between a Primary and Replica dataserver can occur for a number of reasons, including:

- An update or insertion has occurred on the Replica independently of the Primary. This situation can occur if updates are allowed on a Replica that is acting as a read-only Replica for scale out, or in the event of running management or administration scripts on the Replica
- A switch or failover operation has lead to inconsistencies. This can happen if client applications are still writing to the Replica or Primary at the point of the switch.
- A database failure causes a database or table to become corrupted.

When a failure to apply transactions occurs, the problem must be resolved, either by skipping or ignoring the transaction, or fixing and updating the underlying database so that the transaction can be applied.

When a failure occurs, replication is stopped immediately at the first transaction that caused the problem, but it may not be the only transaction and this may require extensive examination of the pending transactions to determine what caused the original database failure and then to fix and address the error and restart replication.

## 7.5.1. Identifying a Transaction Mismatch

When a mismatch occurs, the replicator service will indicate that there was a problem applying a transaction on the Replica. The replication process stops applying changes to the Replica when the first transaction fails to be applied to the Replica. This prevents multiple-statements from failing

When checking the replication status with trepctl, the *pendingError* and *pendingExceptionMessage* will show the error indicating the failure to insert the statement. For example:

```
shell> trepctl status
...
pendingError           : Event application failed: seqno=120 fragno=0 message=java.sql.SQLException: »
    Statement failed on slave but succeeded on master
pendingErrorCode       : NONE
pendingErrorEventId    : mysql-bin.000012:0000000000012967;0
pendingErrorSeqno      : 120
pendingExceptionMessage: java.sql.SQLException: Statement failed on slave but succeeded on master
                         insert into messages values (0,'Trial message','Jack','Jill',now())
...
```

The `trepsvc.log` log file will also contain the error information about the failed statement. For example:

```
...
INFO   | jvm 1    | 2013/06/26 10:14:12 | 2013-06-26 10:14:12,423 [firstcluster -
    q-to-dbms-0] INFO  pipeline.SingleThreadStageTask Performing emergency
    rollback of applied changes
INFO   | jvm 1    | 2013/06/26 10:14:12 | 2013-06-26 10:14:12,424 [firstcluster -
    q-to-dbms-0] INFO  pipeline.SingleThreadStageTask Dispatching error event:
    Event application failed: seqno=120 fragno=0 message=java.sql.SQLException:
    Statement failed on slave but succeeded on master
INFO   | jvm 1    | 2013/06/26 10:14:12 | 2013-06-26 10:14:12,424 [firstcluster -
    pool-2-thread-1] ERROR management.OpenReplicatorManager Received error notification,
    shutting down services :
INFO   | jvm 1    | 2013/06/26 10:14:12 | Event application failed: seqno=120 fragno=0
    message=java.sql.SQLException: Statement failed on slave but succeeded on master
INFO   | jvm 1    | 2013/06/26 10:14:12 | insert into messages values (0,'Trial message',
    'Jack','Jill',now())
INFO   | jvm 1    | 2013/06/26 10:14:12 | com.continuent.tungsten.replicator.applier.ApplierException:
    java.sql.SQLException: Statement failed on slave but succeeded on master
...
```

Once the error or problem has been found, the exact nature of the error should be determined so that a resolution can be identified:

1. Identify the reason for the failure by examining the full error message. Common causes are:

   - Duplicate primary key

     A row or statement is being inserted or updated that already has the same insert ID or would generate the same insert ID for tables that have auto increment enabled. The insert ID can be identified from the output of the transaction using thl. Check the Replica to identify the faulty row. To correct this problem you will either need to skip the transaction or delete the offending row from the Replica dataserver.

     The error will normally be identified due to the following error message when viewing the current replicator status, for example:

     ```
     shell> trepctl status
     ...
     ```

```
pendingError           : Event application failed: seqno=10 fragno=0 »
    message=java.sql.SQLException: Statement failed on slave but succeeded on master
pendingErrorCode       : NONE
pendingErrorEventId    : mysql-bin.000032:0000000000001872;0
pendingErrorSeqno      : 10
pendingExceptionMessage: java.sql.SQLException: Statement failed on slave but succeeded on master
                         insert into myent values (0,'Test Message')
...
```

The error can be generated when an insert or update has taken place on the Replica rather than on the Primary.

To resolve this issue, check the full THL for the statement that failed. The information is provided in the error message, but full examination of the THL can help with identification of the full issue. For example, to view the THL for the sequence number:

```
shell> thl list -seqno 10
SEQ# = 10 / FRAG# = 0 (last frag)
- TIME = 2014-01-09 16:47:40.0
- EPOCH# = 1
- EVENTID = mysql-bin.000032:0000000000001872;0
- SOURCEID = host1
- METADATA = [mysql_server_id=1;dbms_type=mysql;service=firstcluster;shard=test]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- SQL(0) = SET INSERT_ID = 2
- OPTIONS = [##charset = UTF-8, autocommit = 1, sql_auto_is_null = 0, foreign_key_checks = 1, »
     unique_checks = 1, sql_mode = '', character_set_client = 33, collation_connection = 33, »
     collation_server = 8]
- SCHEMA = test
- SQL(1) = insert into myent values (0,'Test Message')
```

In this example, an `INSERT` operation is inserting a new row. The generated insert ID is also shown (in line 9, `SQL(0))...` Check the destination database and determine the what the current value of the corresponding row:

```
mysql> select * from myent where id = 2;
+----+---------------+
| id | msg           |
+----+---------------+
|  2 | Other Message |
+----+---------------+
1 row in set (0.00 sec)
```

The actual row values are different, which means that either value may be correct. In complex data structures, there may be multiple statements or rows that trigger this error if following data also relies on this value.

For example, if multiple rows have been inserted on the Replica, multiple transactions may be affected. In this scenario, checking multiple sequence numbers from the THL will highlight this information.

- Missing table or schema

  If a table or database is missing, this should be reported in the detailed error message. For example:

  ```
  Caused by: java.sql.SQLSyntaxErrorException: Unable to switch to database »
      'contacts'Error was: Unknown database 'contacts'
  ```

  This error can be caused when maintenance has occurred, a table has failed to be initialized properly, or the

- Incompatible table or schema

  A modified table structure on the Replica can cause application of the transaction to fail if there are missing or different column specifications for the table data.

  This particular error can be generated when changes to the table definition have been made, perhaps during a maintenance window.

  Check the table definition on the Primary and Replica and ensure they match.

2. Choose a resolution method:

   Depending on the data structure and environment, resolution can take one of the following forms:

   - Skip the transaction on the Replica

     If the data on the Replica is considered correct, or the data in both tables is the same or similar, the transaction from the Primary to the Replica can be skipped. This process involves placing the replicator online and specifying one or more transactions to be skipped or ignored. At the end of this process, the replicator should be in the `ONLINE` [195] state.

     For more information on skipping single or multiple transactions, see Section 7.5.2, “Skipping Transactions”.

   - Delete the offending row or rows on the Replica

If the data on the Primary is considered canonical, then the data on the Replica can be removed, and the replicator placed online.

> **Warning**
>
> Deleting data on the Replica may cause additional problems if the data is used by other areas of your application, relations to foreign tables.

For example:

```
mysql> delete from myent where id = 2;
Query OK, 1 row affected (0.01 sec)
```

Now place the replicator online and check the status:

```
shell> trepctl online
```

- Restore or reprovision the Replica

  If the transaction cannot be skipped, or the data safely deleted or modified, and only a single Replica is affected, a backup of an existing, working, Replica can be taken and restored to the broken Replica.

  The tungsten_provision_slave command automates this process. See Section 7.6, "Provision or Reprovision a Replica" for more information on reprovisioning.

  To perform a backup and restore, see Section 7.7, "Creating a Backup", or Section 7.8, "Restoring a Backup". To reprovision a Replica from the Primary or another Replica, see tungsten_provision_slave.

## 7.5.2. Skipping Transactions

When a failure caused by a mismatch or failure to apply one or more transactions, the transaction(s) can be skipped. Transactions can either be skipped one at a time, through a specific range, or a list of single and range specifications.

> **Warning**
>
> Skipping over events can easily lead to Replica inconsistencies and later replication errors. Care should be taken to ensure that the transaction(s) can be safely skipped without causing problems. See Section 7.5.1, "Identifying a Transaction Mismatch".

- Skipping a Single Transaction

  If the error was caused by only a single statement or transaction, the transaction can be skipped using trepctl online:

  ```
  shell> trepctl online -skip-seqno 10
  ```

  The individual transaction will be skipped, and the next transaction (11), will be applied to the destination database.

- Skipping a Transaction Range

  If there is a range of statements that need to be skipped, specify a range by defining the lower and upper limits:

  ```
  shell> trepctl online -skip-seqno 10-20
  ```

  This skips all of the transaction within the specified range, and then applies the next transaction (21) to the destination database.

- Skipping Multiple Transactions

  If there are transactions mixed in with others that need to be skipped, the specification can include single transactions and ranges by separating each element with a comma:

  ```
  shell> trepctl online -skip-seqno 10,12-14,16,19-20
  ```

  In this example, only the transactions 11, 15, 17 and 18 would be applied to the target database. Replication would then continue from transaction 21.

Regardless of the method used to skip single or multiple transactions, the status of the replicator should be checked to ensure that replication is online.

# 7.6. Provision or Reprovision a Replica

The command performs three operations automatically:

1. Performs a backup of a remote Replica
2. Copies the backup to the current host
3. Restores the backup

> **Warning**
>
> When using tprovision you must be logged in to the Replica that has failed or that you want to reprovision. You cannot reprovision a Replica remotely.

To use tprovision :

1. Log in to the failed Replica.
2. Select the active Replica within the dataservice that you want to use to reprovision the failed Replica. You may use the Primary but this will impact performance on that host. If you use MyISAM tables the operation will create some locking in order to get a consistent snapshot.
3. Run tprovision specifying the source you have selected:

```
shell> tprovision --source=host2
  NOTE  >> Put alpha replication service offline
  NOTE  >> Create a mysqldump backup of host2 »
  in /opt/continuent/backups/provision_mysqldump_2013-11-21_09-31_52
  NOTE  >> host2 >> Create mysqldump in »
  /opt/continuent/backups/provision_mysqldump_2013-11-21_09-31_52/provision.sql.gz
  NOTE  >> Load the mysqldump file
  NOTE  >> Put the alpha replication service online
  NOTE  >> Clear THL and relay logs for the alpha replication service
```

The default backup service for the host will be used; mysqldump can be used by specifying the `--mysqldump` option.

tprovision handles the cluster status, backup, restore, and repositioning of the replication stream so that restored Replica is ready to start operating again.

> **Important**
>
> When using an Composite Active/Active topology with Tungsten Cluster v5 or earlier, the additional cross-site replicator must also be put offline before restoring data and put online after completion.
>
> ```
> shell> mm_trepctl offline
> shell> tprovision --source=host2
> shell> mm_trepctl online
> shell> mm_trepctl status
> ```

For more information on using tprovision see Section 8.23, “The tprovision Script” .

# 7.7. Creating a Backup

The trepctl backup command backs up a datasource using the default backup tool. During installation, xtrabackup-full will be used if xtrabackup has been installed. Otherwise, the default backup tool used is mysqldump .

> **Important**
>
> For consistency, all backups should include a copy of all `tungsten_SERVICE` schemas. This ensures that when the Tungsten Replicator service is restarted, the correct start points for restarting replication are recorded with the corresponding backup data. Failure to include the `tungsten_SERVICE` schemas may prevent replication from being restart effectively.

Backing up a datasource can occur while the replicator is online:

```
shell> trepctl backup
Backup of dataSource 'host3' succeeded; uri=storage://file-system/store-0000000001.properties
```

By default the backup is created on the local filesystem of the host that is backed up in the `backups` directory of the installation directory. For example, using the standard installation, the directory would be `/opt/continuent/backups` . An example of the directory content is shown below:

```
total 130788
drwxrwxr-x 2 tungsten tungsten      4096 Apr  4 16:09 .
drwxrwxr-x 3 tungsten tungsten      4096 Apr  4 11:51 ..
-rw-r--r-- 1 tungsten tungsten        71 Apr  4 16:09 storage.index
```

```
-rw-r--r-- 1 tungsten tungsten 133907646 Apr  4 16:09 store-0000000001-mysqldump_2013-04-04_16-08_42.sql.gz
-rw-r--r-- 1 tungsten tungsten       317 Apr  4 16:09 store-0000000001.properties
```

For information on managing backup files within your environment, see Section D.1.1, "The `backups` Directory" .

The `storage.index` contains the backup file index information. The actual backup data is stored in the GZipped file. The properties of the backup file, including the tool used to create the backup, and the checksum information, are location in the corresponding `.properties` file. Note that each backup and property file is uniquely numbered so that it can be identified when restoring a specific backup.

A backup can also be initiated and run in the background by adding the & (ampersand) to the command:

```
shell> trepctl backup &
Backup of dataSource 'host3' succeeded; uri=storage://file-system/store-0000000001.properties
```

## 7.7.1. Using a Different Backup Tool

If xtrabackup is installed when the dataservice is first created, xtrabackup will be used as the default backup method. Four built-in backup methods are provided:

- mysqldump — SQL dump to a single file. This is the easiest backup method but it is not appropriate for large data sets.
- xtrabackup — Full backup to a single tar file. This will take longer to take the backup and to restore.
- xtrabackup-full — Full backup to a directory (this is the default if xtrabackup is available and the backup method is not explicitly stated).
- xtrabackup-incremental — Incremental backup from the last xtrabackup-full or xtrabackup-incremental backup.
- mariabackup — Full backup to a single tar file. This will take longer to take the backup and to restore. As of version 6.1.18
- mariabackup-full — Full backup to a directory. As of version 6.1.18
- mariabackup-incremental — Incremental backup from the last mariabackup-full or mariabackup-incremental backup. As of version 6.1.18

The default backup tool can be changed, and different tools can be used explicitly when the backup command is executed. The Percona xtrabackup tool can be used to perform both full and incremental backups. Use of the this tool is optional and can configured during installation, or afterwards by updating the configuration using tpm .

To update the configuration to use xtrabackup , install the tool and then follow the directions for tpm update to apply the `--repl-backup-method=xtrabackup-full` [396] setting.

To use xtrabackup-full without changing the configuration, specify the backup agent to trepctl backup :

```
shell> trepctl backup -backup xtrabackup-full
Backup completed successfully; URI=storage://file-system/store-0000000006.properties
```

## 7.7.2. Using a Different Directory Location

The default backup location the `backups` directory of the Tungsten Cluster installation directory. For example, using the recommended installation location, backups are stored in `/opt/continuent/backups` .

See Section D.1.1.4, "Relocating Backup Storage" for details on changing the location where backups are stored.

## 7.7.3. Creating an External Backup

There are several considerations to take into account when you are using a tool other than Tungsten Cluster to take a backup. We have taken great care to build all of these into our tools. If the options provided do not meet your needs, take these factors into account when taking your own backup.

- How big is your data set?

  The mysqldump tool is easy to use but will be very slow once your data gets too large. We find this happens around 1GB. The xtrabackup tool works on large data sets but requires more expertise. Choose a backup mechanism that is right for your data set.

- Is all of your data in transaction-safe tables?

  If all of your data is transaction-safe then you will not need to do anything special. If not then you need to take care to lock tables as part of the backup. Both mysqldump and xtrabackup take care of this. If you are using other mechanisms you will need to look at stopping the replicator, stopping the database. If you are taking a backup of the Primary then you may need to stop all access to the database.

- Are you taking a backup of the Primary?

  The Tungsten Replicator stores information in a schema to indicate the restart position for replication. On the Primary there can be a slight lag between this position and the actual position of the Primary. This is because the database must write the logs to disk before Tungsten Replicator can read them and update the current position in the schema.

  When taking a backup from the Primary, you must track the actual binary log position of the Primary and start replication from that point after restoring it. See Section 7.8.2, "Restoring an External Backup" for more details on how to do that. When using mysqldump use the `--master-data=2` option. The xtrabackup tool will print the binary log position in the command output.

Using mysqldump can be a very simple way to take consistent backup. Be aware that it can cause locking on MyISAM tables so running it against your Primary will cause application delays. The example below shows the bare minimum for arguments you should provide:

```
shell> mysqldump --opt --single-transaction --all-databases --add-drop-database --master-data=2
```

# 7.8. Restoring a Backup

If a restore is being performed as part of the recovery procedure, consider using the tungsten_provision_slave tool. This will work for restoring from the Primary or a Replica and is faster when you do not already have a backup ready to be restored. For more information, see Section 7.6, "Provision or Reprovision a Replica".

To restore a backup, use the trepctl restore command :

1. Put the replication service offline using trepctl:

   ```
   shell> trepctl offline
   ```

2. Restore the backup using trepctl restore :

   ```
   shell> trepctl restore
   ```

3. Put the replication service online using trepctl:

   ```
   shell> trepctl online
   ```

By default, the restore process takes the latest backup available for the host being restored. Tungsten Cluster *does not* automatically locate the latest backup within the dataservice across all datasources.

## 7.8.1. Restoring a Specific Backup

To restore a specific backup, specify the location of the corresponding properties file using the format:

```
storage://storage-type/location
```

For example, to restore the backup from the filesystem using the information in the properties file `store-0000000004.properties` , login to the failed host:

1. Put the replication service offline using trepctl:

   ```
   shell> trepctl offline
   ```

2. Restore the backup using trepctl restore:

   ```
   shell> trepctl restore \
   -uri storage://file-system/store-0000000004.properties
   ```

3. Put the replication service online using trepctl:

   ```
   shell> trepctl online
   ```

## 7.8.2. Restoring an External Backup

If a backup has been performed outside of Tungsten Cluster, for example from filesystem snapshot or a backup performed outside of the dataservice, follow these steps:

1. Put the replication service offline using trepctl:

   ```
   shell> trepctl offline
   ```

2. Reset the THL, either using thl or by deleting the files directly :

```
shell> thl -service alpha purge
```

3. Restore the data or files using the external tool. This may require the database server to be stopped. If so, you should restart the database server before moving to the next step.

   **Note**

   The backup must be complete and the `tungsten` specific schemas must be part of the recovered data, as they are required to restart replication at the correct point. See Section 7.7.3, "Creating an External Backup" for more information on creating backups.

4. There is some additional work if the backup was taken of the Primary server. There may be a difference between the binary log position of the Primary and what is represented in the `trep_commit_seqno`. If these values are the same, you may proceed without further work. If not, the content of `trep_commit_seqno` must be updated.

   - Retrieve the contents of `trep_commit_seqno` :

     ```
     shell> echo "select seqno,source_id, eventid from tungsten_alpha.trep_commit_seqno" | tpm mysql
     seqno source_id eventid
     32033674 host1 mysql-bin.000032:0000000473860407;-1
     ```

   - Compare the results to the binary log position of the restored backup. For this example we will assume the backup was taken at `mysql-bin.000032:473863524`. Return to the Primary and find the correct sequence number for that position :

     ```
     shell> ssh host1

     shell> thl dsctl -event mysql-bin.000032:0000000473863524
     dsctl -service alpha set -reset -seqno 7748 -epoch 0 -event-id "mysql-bin.000032:0000000473863524" -source-id "db1-east.continuent.com"

     ~OR~

     shell> thl list -event mysql-bin.000032:0000000473863524 -headers
     SEQ# = 7748 / FRAG# = 0 (last frag)
     - FILE = thl.data.0000000010
     - TIME = 2014-10-17 16:58:11.0
     - EPOCH# = 0
     - EVENTID = mysql-bin.000032:0000000473863524;-1
     - SOURCEID = db1-east.continuent.com

     shell> exit
     ```

   - Return to the Replica node and run dsctl set to update the `trep_commit_seqno` table :

     ```
     shell> dsctl -service alpha set -reset \
          -seqno 7748 \
          -epoch 0 \
          -source-id db1-east.continuent.com \
          -event-id mysql-bin.000032:0000000473863524
     ```

5. Put the replication service online using trepctl:

   ```
   shell> trepctl online
   ```

## 7.8.3. Restoring from Another Replica

If a restore is being performed as part of the recovery procedure, consider using the tungsten_provision_slave tool. This is will work for restoring from the Primary or a Replica and is faster if you do not already have a backup ready to be restored. For more information, see Section 7.6, "Provision or Reprovision a Replica".

Data can be restored to a Replica by performing a backup on a different Replica, transferring the backup information to the Replica you want to restore, and then running restore process.

For example, to restore the `host3` from a backup performed on `host2` :

1. Run the backup operation on `host2` :

   ```
   shell> trepctl backup
   Backup of dataSource 'host2' succeeded; uri=storage://file-system/store-0000000006.properties
   ```

2. Copy the backup information from `host2` to `host3`. See Section D.1.1.3, "Copying Backup Files" for more information on copying backup information between hosts. If you are using xtrabackup there will be additional files needed before the next step. The example below uses scp to copy a mysqldump backup:

   ```
   shell> cd /opt/continuent/backups
   ```

```
shell> scp store-[0]*6[\.-]* host3:$PWD/
store-0000000006-mysqldump-812096863445699665.sql                    100%  234MB  18.0MB/s   00:13
store-0000000006.properties                                          100%  314     0.3KB/s   00:00
```

If you are using xtrabackup:

```
shell> cd /opt/continuent/backups/xtrabackup
shell> rsync -aze ssh full_xtrabackup_2014-08-16_15-44_86 host3:$PWD/
```

3. Put the replication service offline using trepctl:

```
shell> trepctl offline
```

4. Restore the backup using trepctl restore :

```
shell> trepctl restore
```

> **Note**
>
> Check the ownership of files if you have trouble transferring files or restoring the backup. They should be owned by the Tungsten system user to ensure proper operation.

5. Put the replication service online using trepctl:

```
shell> trepctl online
```

## 7.8.4. Manually Recovering from Another Replica

In the event that a restore operation fails, or due to a significant failure in the dataserver, an alternative option is to seed the failed dataserver directly from an existing running Replica.

For example, on the host `host2` , the data directory for MySQL has been corrupted, and mysqld will no longer start. This status can be seen from examining the MySQL error log in `/var/log/mysql/error.log:`

```
130520 14:37:08 [Note] Recovering after a crash using /var/log/mysql/mysql-bin
130520 14:37:08 [Note] Starting crash recovery...
130520 14:37:08 [Note] Crash recovery finished.
130520 14:37:08 [Note] Server hostname (bind-address): '0.0.0.0'; port: 13306
130520 14:37:08 [Note]   - '0.0.0.0' resolves to '0.0.0.0';
130520 14:37:08 [Note] Server socket created on IP: '0.0.0.0'.
130520 14:37:08 [ERROR] Fatal error: Can't open and lock privilege tables: Table 'mysql.host' doesn't exist
130520 14:37:08 [ERROR] /usr/sbin/mysqld: File '/var/run/mysqld/mysqld.pid' not found (Errcode: 13)
130520 14:37:08 [ERROR] /usr/sbin/mysqld: Error reading file 'UNKNOWN' (Errcode: 9)
130520 14:37:08 [ERROR] /usr/sbin/mysqld: Error on close of 'UNKNOWN' (Errcode: 9)
```

Performing a restore operation on this Replica may not work. To recover from another running Replica, `host3` , the MySQL data files can be copied over to `host2` directly using the following steps:

1. Put the `host2` replication service offline using trepctl:

```
shell> trepctl offline
```

2. Put the `host3` replication service offline using trepctl:

```
shell> trepctl offline
```

3. Stop the mysqld service on `host2`:

```
shell> sudo /etc/init.d/mysql stop
```

4. Stop the mysqld service on `host3`:

```
shell> sudo /etc/init.d/mysql stop
```

5. Delete the mysqld data directory on `host2` :

```
shell> sudo rm -rf /var/lib/mysql/*
```

6. If necessary, ensure the `tungsten` user can write to the MySQL directory:

```
shell> sudo chmod 777 /var/lib/mysql
```

7. Use rsync on `host3` to send the data files for MySQL to `host2` :

```
shell> rsync -aze ssh /var/lib/mysql/* host2:/var/lib/mysql/
```

You should synchronize all locations that contain data. This includes additional folders such as `innodb_data_home_dir` or `innodb_log_group_home_dir`. Check the `my.cnf` file to ensure you have the correct paths.

Once the files have been copied, the files should be updated to have the correct ownership and permissions so that the Tungsten service can read them.

8. Start the mysqld service on `host3` :

```
shell> sudo /etc/init.d/mysql start
```

9. Put the `host3` replication service online using trepctl:

```
shell> trepctl online
```

10. Update the ownership and permissions on the data files on `host2`:

```
host2 shell> sudo chown -R mysql:mysql /var/lib/mysql
host2 shell> sudo chmod 770 /var/lib/mysql
```

11. Clear out the THL files on the target node `host2` so the Replica replicator service may start cleanly:

```
host2 shell> thl purge
```

12. Start the mysqld service on `host2` :

```
shell> sudo /etc/init.d/mysql start
```

13. Put the `host2` replication service online using trepctl:

```
shell> trepctl online
```

## 7.8.5. Reprovision a MySQL Replica using rsync

The steps below will guide you through the process of restoring a MySQL Replica node by using rsync.

The following process has the following caveats:

- You can sustain downtime on the node used as a source.
- You can either ssh between hosts as root, or have root level access to temporarily change file ownership.

Steps

1. Establish `ssh` as root between hosts, or if you can't set up ssh as root, on the target host, `chown` ownership of mysql target directories to tungsten, then run rysnc as command in Step 3 as the tungsten user.
2. On failed host:
   - Shut down mysql if it is still running.
   - Determine all mysql data directories (datadir, binary log dir, etc) e.g. `/var/lib/mysql`
   - Clear out database files from the failed host.

     ```
     shell> rm -rf /var/lib/mysql/*
     ```
3. On Source host:
   - Shut down mysql.
   - Use rsync to copy the datafiles to the target (specify 'z' if CPU available for compression)

     ```
     sourcehost> rsync -avz --progress /source/dir/ targetHost:/target/dir/
     ```
4. Wait for the rsync to complete.
5. On Source host:
   - Restart mysql.
   - Bring the replicator online.

```
shell> trepctl online
```

- Wait for replication to cacth up.

6. On restored, target, host:

   - Fix the ownership on ALL data directories, e.g.

```
shell> chown -R mysql: /var/lib/mysql
```

   - Start mysql.
   - Bring the replicator online.

```
shell> trepctl online
```

   - Wait for replication to cacth up.

# 7.9. Deploying Automatic Replicator Recovery

Automatic recovery enables the replicator to go back `ONLINE` [195] in the event of a transient failure that is triggered during either the `ONLINE` [195] or `GOING-ONLINE:SYNCHRONIZING` [195] state that would otherwise trigger a change of states to `OFFLINE` [195]. For example, connection failures, or restarts in the MySQL service, trigger the replicator to go `OFFLINE` [195]. With autorecovery enabled, the replicator will attempt to put the replicator `ONLINE` [195] again to keep the service running. Failures outside of these states will not trigger autorecovery.

Autorecovery operates by scheduling an attempt to go back online after a transient failure. If autorecovery is enabled, the process works as follows:

1. If a failure is identified, the replicator attempts to go back online after a specified delay. The delay allows the replicator time to decide whether autorecovery should be attempted. For example, if the MySQL server restarts, the delay gives time for the MySQL server to come back online before the replicator goes back online.

2. Recovery is attempted a configurable number of times. This presents the replicator from continually attempting to go online within a service that has a more serious failure. If the replicator fails to go `ONLINE` [195] within the configurable reset interval, then the replicator will go to the `OFFLINE` [195] state.

3. If the replicator remains in the `ONLINE` [195] state for a configurable period of time, then the automatic recovery is deemed to have succeeded. If the autorecovery fails, then the autorecovery attempts counter is incremented by one.

The configurable parameters are set using tpm within the static properties for the replicator:

- `--auto-recovery-max-attempts` [395]

  Sets the maximum number of attempts to automatically recovery from any single failure trigger. This prevents the autorecovery mechanism continually attempting autorecover. The current number of attempts is reset if the replicator remains online for the configured reset period.

- `--auto-recovery-delay-interval` [395]

  The delay between entering the `OFFLINE` [195] state, and attempting autorecovery. On servers that are busy, use some form of network or HA solution, or have high MySQL restart/startup times, this value should be configured accordingly to give the underlying services time to startup again after failure.

- `--auto-recovery-reset-interval` [396]

  The duration after a successful autorecovery has been completed that the replicator must remain in the `ONLINE` [195] state for the recovery process to be deemed to have succeeded. The number of attempts for autorecovery is reset to 0 (zero) if the replicator stays up for this period of time.

Auto recovery is enabled only when the `--auto-recovery-max-attempts` [395] parameter is set to a non-zero value.

To enable:

```
shell> tpm update alpha --auto-recovery-max-attempts=5
```

The autorecovery status can be monitored within `trepsvc.log` and through the *`autoRecoveryEnabled`* and *`autoRecoveryTotal`* parameters output by trepctl. For example:

```
shell> trepctl status
Processing status command...
```

```
NAME                    VALUE
----                    -----
...
autoRecoveryEnabled    : false
autoRecoveryTotal      : 0
...
```

The above output indicates that the autorecovery service is disabled. The `autoRecoveryTotal` is a count of the number of times the autorecovery has been completed since the replicator has started.

# 7.10. Migrating and Seeding Data

## 7.10.1. Migrating from MySQL Native Replication 'In-Place'

If you are migrating an existing MySQL native replication deployment to use Tungsten Cluster or the standalone Tungsten Replicator the configuration of the must be updated to match the status of the Replica.

1. Deploy Tungsten Replicator using the model or system appropriate according to Chapter 2, *Deployment Overview*. Ensure that the Tungsten Cluster is not started automatically by excluding the `--start` [424] or `--start-and-report` [424] options from the tpm commands.

2. On each Replica

   Confirm that native replication is working on all Replica nodes :

   ```
   shell> echo 'SHOW SLAVE STATUS\G' | tpm mysql | \
   egrep ' Master_Host| Last_Error| Slave_SQL_Running'
                   Master_Host: tr-ssl1
             Slave_SQL_Running: Yes
                    Last_Error:
   ```

3. On the Primary and each Replica

   Reset the Tungsten Replicator position on all servers :

   ```
   shell> replicator start offline
   shell> trepctl -service alpha reset -all -y
   ```

4. On the Primary

   Start Tungsten Replicator :

   ```
   shell> replicator start
   ```

5. On each Replica

   Record the current Replica log position (as reported by the `Master_Log_File` and `Exec_Master_Log_Pos` output from `SHOW SLAVE STATUS`. Ideally, each Replica should be stopped at the same position:

   ```
   shell> echo 'SHOW SLAVE STATUS\G' | tpm mysql | \
   egrep ' Master_Host| Last_Error| Master_Log_File| Exec_Master_Log_Pos'
                   Master_Host: tr-ssl1
               Master_Log_File: mysql-bin.000025
                    Last_Error: Error executing row event: 'Table 'tungsten_alpha.heartbeat' doesn't exist'
          Exec_Master_Log_Pos: 181268
   ```

   If you have multiple Replicas configured to read from this Primary, record the Replica position individually for each host. Once you have the information for all the hosts, determine the earliest log file and log position across all the Replicas, as this information will be needed when starting replication. If one of the servers does not show an error, it may be replicating from an intermediate server. If so, you can proceed normally and assume this server stopped at the same position as the host is replicating from.

6. On the Primary

   Take the replicator offline and clear the THL:

   ```
   shell> trepctl offline
   shell> trepctl -service alpha reset -all -y
   ```

7. On the Primary

   Start replication, using the *lowest* binary log file and log position from the Replica information determined previously.

   ```
   shell> trepctl online -from-event 000025:181268
   ```

Tungsten Replicator will start reading the MySQL binary log from this position, creating the corresponding THL event data.

8. On each Replica

   a. Disable native replication to prevent native replication being accidentally started on the Replica.

      *On MySQL 5.0 or MySQL 5.1:*

      ```
      shell> echo "STOP SLAVE; CHANGE MASTER TO MASTER_HOST='';" | tpm mysql
      ```

      *On MySQL 5.5 or later:*

      ```
      shell> echo "STOP SLAVE; RESET SLAVE ALL;" | tpm mysql
      ```

   b. If the final position of MySQL replication matches the lowest across all Replicas, start Tungsten Replicator services :

      ```
      shell> trepctl online
      ```

      The Replica will start reading from the binary log position configured on the Primary.

9. Check that replication is operating correctly by using trepctl status on the Primary and each Replica to confirm the correct position.

10. Remove the `master.info` file on each Replica to ensure that when a Replica restarts, it does not connect up to the Primary MySQL server again.

Once these steps have been completed, Tungsten Replicator should be operating as the replication service for your MySQL servers. Use the information in Chapter 7, *Operations Guide* to monitor and administer the service.

## 7.10.2. Seeding Data for Heterogeneous Replication

Seeding data for heterogeneous targets is a complex process that can have many challenges that all depend on the target and the amount of data requiring seeding.

The steps outlined below come with their own challenges that may prove not to be suitable in your own environment. Therefore, any pre-seeding process needs to be fully understood and evaluated

The following requirements are needed for this process:

- A temporary host meeting all required pre-requisites, with an empty MySQL instance matching the same version as the original source, and with the default storage engine set to `BLACKHOLE`
- Enough disk space on the temporary host to hold binary logs and THL logs equivalent to the amount of data being seeded
- The ability to extract the data from the source, using mysqldump, or via reverse engineering SQL Statements

The process in summary is as follows:

- Configure replicator to extract from the temporary instance
- Extract data from the source capturing the binlog position appropriate to the export
- Import the data into the temporary instance and allow the replicator to load the target
- Re-Configure the replicator to extract from the original source, positioned to start from the co-ordinates noted during the export

### 7.10.2.1. Seeding Data from a Standalone Source

Step 1: Configure the Temporary Instance

- Build an empty MySQL Instance providing that all the pre-requisites are in place. These are outline in Appendix B, *Prerequisites*
- To ensure compatibility, you need to make sure that the version of MySQL used matches the version of MySQL running on the main source.
- Once the instance is running, you need to pre-create all of the tables that you will be loading. This must be done manually as you need to ensure that each table is created with the `ENGINE=BLACKHOLE` option
- The use of the `BLACKHOLE` engine will mean that the data doesn't actually get stored when written to the database, however binary logs are generated and it is these that the replicator requires

Step 2: Configure the Replicator

- Follow the steps outlined in Section 3.2, "Deploying a Primary/Replica Topology" to configure an extractor against the Temporary host.
- Ensure that the configuration includes the following entries to enable heterogeneous replication, and ensure you qualify the objects (schemas and/or tables) that you want to seed

```
enable-heterogeneous-master=true
svc-extractor-filters=replicate
property=replicator.filter.replicate.do=schema.table
```

- Once installed, start the replicator

Step 3: Build the Target Schema

- Depending on the target, you may need to pre-create the final objects in your target environment. Use `ddlscan` to do this now if this is required

Step 4: Configure the Applier

- Ensure the applier host meets all the required pre-requisites and then configure the applier appropriate to the target you are applying to.
- Follow the appropriate steps in Chapter 4, *Deploying Appliers* to configure a standalone applier, ensuring that it is configured to connect to the temporary extractor installed in Step 2
- Once configured, start the applier. At this stage you should not see any replication traffic since the temporary host will have no data written to it

Step 5: Export the data from the source

- We now need to export the data from the source.
- Using mysqldump we need to ensure we capture the binlog position and we also need to ensure that the export does NOT contain DB or TABLE DDL
- The following example can be used as a template. In this example we are exporting an entire schema. Use the appropriate options if you require only specific tables

```
mysqldump -u root -psecret -B hr --no-create-db --no-create-info --master-data=2 >dump.sql
```

Step 6: Import the Data

- Now that we have the data we can import this into the Temporary Instance
- As you load the data, you can monitor replication and you should see the data loading into your target envrionment.
- Providing you created the tables correctly with the `BLACKHOLE` engine, you should see that a `select count` on the tables in the temporary instance should return a row count of zero.
- When the load has finished and the applier has completed replication, stop both replicators using the following command:

```
shell> replicator stop
```

  We have now finished with the temporary MySQL instance

Step 7: Install Extractor from Main Source Host

- Follow the steps outlined in Section 3.2, "Deploying a Primary/Replica Topology" to configure an extractor against the Source host.
- Ensure that the configuration includes the following entries to enable heterogeneous replication, and ensure you qualify the objects (schemas and/or tables) as required, for example

```
enable-heterogeneous-master=true
svc-extractor-filters=replicate
property=replicator.filter.replicate.do=schema.table
```

- Once installed, start the replicator in an offline state

```
shell> replicator start offline
```

Step 8: Reconfigure the Applier

- We now need to reconfigure the applier, but first we need to uninstall the software to ensure we have a clean build and any THL from the pre-load has been cleared

```
shell> cd /opt/continuent/software/tungsten-replicator-7.1.2-81
shell> tools/tpm uninstall --i-am-sure
```

- Place the correct configuration into the `/etc/tungsten/tungsten.ini` file ensuring that `start-and-report=false` is set and that the applier is now configured to point to the main extractor
- Install the software

```
shell> cd /opt/continuent/software/tungsten-replicator-7.1.2-81
shell> tools/tpm install
```

- Finally start the repicator in an offline state, and issue a reset to be sure the previous tracking schema is clean:

```
shell> replicator start offline
shell> trepctl -service servicename reset -all -y
```

Step 9: Position the Replicator

- The final step is to now position the extractor to pick up from the position that the export in step 5 was taken.
- Locate the dump file and issue the following command:

```
shell> grep "CHANGE MASTER" dump.sql
-- CHANGE MASTER TO MASTER_LOG_FILE='mysql-bin.000003', MASTER_LOG_POS=847;
```

  Taking the sequence value from MASTER_LOG_FILE filename and the MASTER_LOG_POS, issue the following statement on the EXTRACTOR host:

```
shell> trepctl online -from-event 000003:847
```

- Once the command has completed, the extractor will be online, you can now bring the applier online

```
shell> trepctl online
```

## 7.10.2.2. Seeding Data from a Cluster, for a Cluster-Extractor Target

Step 1: Configure the Temporary Instance

- Build an empty MySQL Instance providing that all the pre-requisites are in place. These are outline in Appendix B, *Prerequisites*
- To ensure compatibility, you need to make sure that the version of MySQL used matches the version of MySQL running on the main source.
- Once the instance is running, you need to pre-create all of the tables that you will be loading. This must be done manually as you need to ensure that each table is created with the `ENGINE=BLACKHOLE` option
- The use of the `BLACKHOLE` engine will mean that the data doesn't actually get stored when written to the database, however binary logs are generated and it is these that the replicator requires

Step 2: Configure the Replicator

- Follow the steps outlined in Section 3.2, "Deploying a Primary/Replica Topology" to configure an extractor against the Temporary host.
- Ensure that the configuration includes the following entries to enable heterogeneous replication, and ensure you qualify the objects (schemas and/or tables) that you want to seed

```
enable-heterogeneous-master=true
svc-extractor-filters=replicate
property=replicator.filter.replicate.do=schema.table
```

- Once installed, start the replicator

Step 3: Build the Target Schema

- Depending on the target, you may need to pre-create the final objects in your target environment. Use `ddlscan` to do this now if this is required

Step 4: Configure the Applier

In a Cluster-Extractor environment, we will use the same applier after we have seeded the target.

- Ensure the applier host meets all the required pre-requisites and then configure the applier appropriate to the target you are applying to.

- Follow the appropriate steps in Chapter 4, *Deploying Appliers* to configure a standalone applier, ensuring that it is configured to connect to the temporary extractor installed in Step 2. At this stage do not follow the Cluster-Extractor setup

  > **Important**
  >
  > When setting the service name for this temporary seeding process, ensure the servicename you choose matches the servicename of the main source cluster that we will later connect to for normal operation

- Once configured, start the applier. At this stage you should not see any replication traffic since the temporary host will have no data written to it

Step 5: Export the data from the source

- We now need to export the data from the source. There are two ways to do this with a cluster. We can either take an export from the Primary, or we an take an export from a Replica.
- Export from a Primary:
  - If you are able to export from the Primary, then using mysqldump we need to ensure we capture the binlog position and we also need to ensure that the export does NOT contain DB or TABLE DDL
  - The following example can be used as a template. In this example we are exporting an entire schema. Use the appropriate options if you require only specific tables

    ```
    mysqldump -u root -psecret -B hr --no-create-db --no-create-info --master-data=2 >dump.sql
    ```

- Export from a Replica:
  - To export from a Replica, we cannot obtain the binlog position correctly as the one we need is specific to the Primary, however, exporting from a Replica means that we can utilise the features of Tungsten Cluster to isolate the node
  - First, set the cluster to MAINTENANCE mode, and then SHUN the node that we wish to export from

    ```
    shell> cctrl
    cctrl> set policy maintenance
    cctrl> datasource Replicahost shun
    ```

  - Next, we can take the export

    ```
    mysqldump -u root -psecret -B hr --no-create-db --no-create-info >dump.sql
    ```

  - The final step is to capture the current replication position using dsctl

    ```
    shell> dsctl get -ascmd
    dsctl set -seqno 9 -epoch 2 -event-id "mysql-bin.000003:0000000000002608;-1" -source-id "db1"
    ```

    Make a note of the output from running this command as we will need it later

  - Finally, you can re-introduce the node back into the cluster

    ```
    shell> cctrl
    cctrl> datasource Replicahost recover
    cctrl> set policy automatic
    ```

Step 6: Import the Data

- Now that we have the data we can import this into the Temporary Instance
- As you load the data, you can monitor replication and you should see the data loading into your target envrionment.
- Providing you created the tables correctly with the `BLACKHOLE` engine, you should see that a `select count` on the tables in the temporary instance should return a row count of zero.
- When the load has finished and the applier has completed replication, stop both replicators using the following command:

  ```
  shell> replicator stop
  ```

  We have now finished with the temporary MySQL instance

Step 7: Reconfigure the Applier as a Cluster-Extractor

- We now need to reconfigure the applier as a cluster Replica, but first we need to uninstall the software to ensure we have a clean build and any THL from the pre-load has been cleared

```
shell> cd /opt/continuent/software/tungsten-replicator-7.1.2-81
shell> tools/tpm uninstall --i-am-sure
```

- Place the correct configuration into the `/etc/tungsten/tungsten.ini` file ensuring that `start-and-report=false` is set
- Install the software

```
shell> cd /opt/continuent/software/tungsten-replicator-7.1.2-81
shell> tools/tpm install
```

- Finally start the repicator in an offline state:

```
shell> replicator start offline
```

Step 8: Position the Replicator

- The final step is to now position the replicator to pick up from the position that the export in step 5 was taken.
- If you took the export from the Replica and already have a dsctl set command, then move onto the next step. If you took an export from the Primary then we need to retrieve the correct positions from the dump file

  Locate the dump file and issue the following command:

```
shell> grep "CHANGE MASTER" dump.sql
-- CHANGE MASTER TO MASTER_LOG_FILE='mysql-bin.000003', MASTER_LOG_POS=847;
```

  Taking the sequence value from `MASTER_LOG_FILE` filename and the `MASTER_LOG_POS`, issue the following statement on the Primary host in the cluster, and copy the resulting dsctl command:

```
shell> tungsten_find_position mysql-bin.000003:847
dsctl set -reset -seqno 9 -epoch 2 -event-id "mysql-bin.000003:0000000000000847;-1" -source-id "db1"
```

- Taking the dsctl that you obtained either from the Replica in Step 5 or from the steps just above, run the dsctl command on the applier host ONLY
- Once the command has completed, you can now bring the replicator online

```
shell> replicator start offline
```

# 7.11. Switching Primary Hosts

In the event of a failure, or during the process of performing maintenance on a running cluster, the roles of the Extractor and Appliers within the cluster may need to be swapped.

The basic sequence of operation for switching Primary and Replicas is:

1. Switch Replicas to offline state
2. Switch Primary to offline status
3. Set an existing Replica to have the *`master`* role
4. Set each Replica with the *`slave`* role, updating the Extractor URI (where the THL logs will be loaded) to the new Extractor host
5. Switch the new $ctpri; to online state
6. Switch the new $ctreps; to online state

Depending on the situation when the switch is performed, the switch can be performed either without waiting for the hosts to be synchronized (i.e. in a failure situation), or by explicitly waiting for the Replica that will be promoted to the Primary role.

To perform an ordered switch of the Primary. In the example below, Primary host `host1` will be switched to `host3`, and the remaining hosts (`host1` and `host2`) will be configured as Replicas to the new Primary:

1. If you are performing the switch as part of maintenance or other procedures, you should perform a safe switch, ensuring the Replicas are up to date with the Primary:

   a. Synchronize the database and the transaction history log. This will ensure that the two are synchronized, and provide you with a sequence number to ensure the Replicas are up to date:

```
shell> trepctl -host host1 flush
```

```
Master log is synchronized with database at log sequence number: 1405
```

Keep a note of the sequence number.

b. For each current Replica within the cluster, wait until the Primary sequence number has been reached, and then put the Replica into the offline state:

```
shell> trepctl -host host2 wait -applied 1405
shell> trepctl -host host2 offline
shell> trepctl -host host3 wait -applied 1405
shell> trepctl -host host3 offline
```

If the Primary has failed, or once the Replicas and Primaries are in sync, you can perform the remainder of the steps to execute the physical switch.

2. Switch the Primary to the offline state:

```
shell> trepctl -host host1 offline
```

3. Configure the new designated Primary to the *Primary* role:

```
shell> trepctl -host host3 setrole -role master
```

Switch the Primary to the online state:

```
shell> trepctl -host host3 online
```

4. For each Replica, set the role to Replica, supplying the URI of the THL service on the Primary:

```
shell> trepctl -host host1 setrole -role slave -uri thl://host3:2112
```

In the above example we are using the default THL port (2112).

Put the new Replica into the online state:

```
shell> trepctl -host host1 online
```

Repeat for the remaining Replicas:

```
shell> trepctl -host host2 setrole -role slave -uri thl://host3:2112
shell> trepctl -host host2 online
```

Once completed, the state of each host can be checked to confirm that the switchover has completed successfully:

```
appliedLastEventId     : mysql-bin.000005:0000000000002100;0
appliedLastSeqno       : 1405
appliedLatency         : 0.094
dataServerHost         : host1
masterConnectUri       : thl://host3:2112
role                   : slave
state                  : ONLINE
-----
appliedLastEventId     : mysql-bin.000005:0000000000002100;0
appliedLastSeqno       : 1405
appliedLatency         : 0.149
dataServerHost         : host2
masterConnectUri       : thl://host3:2112
role                   : slave
state                  : ONLINE
-----
appliedLastEventId     : mysql-bin.000005:0000000000002100;0
appliedLastSeqno       : 1405
appliedLatency         : 0.061
dataServerHost         : host3
masterConnectUri       : thl://host1:2112/
role                   : master
state                  : ONLINE
```

In the above, `host1` and `host2` are now getting the THL information from `host1`, with each acting as a Replica to the `host1` as Primary.

## 7.12. Configuring Parallel Replication

The replication stream within MySQL is by default executed in a single-threaded execution model. Using Tungsten Replicator, the application of the replication stream can be applied in parallel. This improves the speed at which the database is updated and helps to reduce the effect of Replicas lagging behind the Primary which can affect application performance. Parallel replication operates by distributing the events from the replication stream from different database schemas in parallel on the Replica. All the events in one schema are applied in sequence, but

events in multiple schemas can be applied in parallel. Parallel replication will not help in those situations where transactions operate across schema boundaries.

Parallel replication supports two primary options:

- Number of parallel channels — this configures the maximum number of parallel operations that will be performed at any one time. The number of parallel replication streams should match the number of different schemas in the source database, although it is possible to exhaust system resources by configuring too many. If the number of parallel threads is less than the number of schemas, events are applied in a round-robin fashion using the next available parallel stream.
- Parallelization type — the type of parallelization to be employed. The disk method is the recommended solution.

Parallel replication can be enabled during installation by setting the appropriate options during the initial configuration and installation. To enable parallel replication after installation, you must configure each host as follows:

1. Put the replicator offline:

```
shell> trepctl offline
```

2. Reconfigure the replication service to configure the parallelization:

```
shell> tpm update firstrep --host=host2 \
    --channels=5 --svc-parallelization-type=disk
```

3. Then restart the replicator to enable the configuration:

```
shell> replicator restart
Stopping Tungsten Replicator Service...
Stopped Tungsten Replicator Service.
Starting Tungsten Replicator Service...
```

The current configuration can be confirmed by checking the *channels* configured in the status information:

```
shell> trepctl status
Processing status command...
NAME                     VALUE
----                     -----
appliedLastEventId     : mysql-bin.000005:0000000000004263;0
appliedLastSeqno       : 1416
appliedLatency         : 1.0
channels               : 5
...
```

More detailed information can be obtained by using the trepctl status -name stores command, which provides information for each of the parallel replication queues:

```
shell> trepctl status -name stores
Processing status command (stores)...
NAME                      VALUE
----                      -----
activeSeqno             : 0
doChecksum              : false
flushIntervalMillis     : 0
fsyncOnFlush            : false
logConnectionTimeout    : 28800
logDir                  : /opt/continuent/thl/firstrep
logFileRetainMillis     : 604800000
logFileSize             : 100000000
maximumStoredSeqNo      : 1416
minimumStoredSeqNo      : 0
name                    : thl
readOnly                : false
storeClass              : com.continuent.tungsten.replicator.thl.THL
timeoutMillis           : 2147483647
NAME                      VALUE
----                      -----
criticalPartition       : -1
discardCount            : 0
estimatedOfflineInterval: 0.0
eventCount              : 0
headSeqno               : -1
intervalGuard           : AtomicIntervalGuard (array is empty)
maxDelayInterval        : 60
maxOfflineInterval      : 5
maxSize                 : 10
name                    : parallel-queue
queues                  : 5
serializationCount      : 0
serialized              : false
stopRequested           : false
```

```
store.0                 : THLParallelReadTask task_id=0 thread_name=store-thl-0 »
    hi_seqno=0 lo_seqno=0 read=0 accepted=0 discarded=0 events=0
store.1                 : THLParallelReadTask task_id=1 thread_name=store-thl-1 »
    hi_seqno=0 lo_seqno=0 read=0 accepted=0 discarded=0 events=0
store.2                 : THLParallelReadTask task_id=2 thread_name=store-thl-2 »
    hi_seqno=0 lo_seqno=0 read=0 accepted=0 discarded=0 events=0
store.3                 : THLParallelReadTask task_id=3 thread_name=store-thl-3 »
    hi_seqno=0 lo_seqno=0 read=0 accepted=0 discarded=0 events=0
store.4                 : THLParallelReadTask task_id=4 thread_name=store-thl-4 »
    hi_seqno=0 lo_seqno=0 read=0 accepted=0 discarded=0 events=0
storeClass              : com.continuent.tungsten.replicator.thl.THLParallelQueue
syncInterval            : 10000
Finished status command (stores)...
```

To examine the individual threads in parallel replication, you can use the trepctl status -name shards status option, which provides information for each individual shard thread:

```
Processing status command (shards)...
NAME                VALUE
----                -----
appliedLastEventId: mysql-bin.000005:0000000013416909;0
appliedLastSeqno  : 1432
appliedLatency    : 0.0
eventCount        : 28
shardId           : cheffy
stage             : q-to-dbms
...
Finished status command (shards)...
```

# 7.13. Performing Database or OS Maintenance

When performing database or operating system maintenance, datasources should be temporarily disabled by placing them into the `OFFLINE` [195] state. For maintenance operations on a Primary, the current Primary should be switched, the required maintenance steps performed, and then the Primary switched back. Detailed steps are provided below for different scenarios.

## 7.13.1. Performing Maintenance on a Single Replica

To perform maintenance on a single Replica, you should ensure that your application is not using the Replica, perform the necessary maintenance, and then re-enable the Replica within your application.

The steps are:

1. Put the replicator into the offline state to prevent replication and changes being applied to the database:

   ```
   shell> trepctl -host host1 offline
   ```

   To perform operating system maintenance, including rebooting the system, the replicator can be stopped completely:

   ```
   shell> replicator stop
   ```

2. Perform the required maintenance, including updating the operating system, software or hardware changes.

3. Validate the server configuration:

   ```
   shell> tpm validate
   ```

4. Put the replicator back online:

   ```
   shell> trepctl -host host1 online
   ```

   Or if you have stopped the replicator, restart the service again:

   ```
   shell> replicator start
   ```

Once the datasource is back online, monitor the status of the service and ensure that the replicator has started up and that transactions are being extracted or applied.

## 7.13.2. Performing Maintenance on a Primary

Maintenance, including MySQL admin or schema updates, should not be performed directly on a Primary as this may upset the replication and therefore availability and functionality of the Replicas which are reading from the Primary.

To effectively make the modifications, you should switch the Primary host, then operate on the Primary as if it were Replica, removing it from the replicator service configuration. This helps to minimize any problems or availability that might be cause by performing operations directly on the Primary.

The complete sequence and commands required to perform maintenance on an active Primary are shown in the table below. The table assumes a dataservice with three datasources:

| Step | Description | Command | host1 | host2 | host3 |
|---|---|---|---|---|---|
| 1 | Initial state | | Primary | Replica | Replica |
| 2 | Switch Primary to `host2` | See Section 7.11, "Switching Primary Hosts" | Replica | Primary | Replica |
| 3 | Put Replica into OFFLINE state | trepctl -host host1 offline | Offline | Primary | Replica |
| 4 | Perform maintenance | | Offline | Primary | Replica |
| 5 | Validate the `host1` server configuration | tpm validate | Offline | Primary | Replica |
| 6 | Put the Replica online | trepctl -host host1 online | Replica | Primary | Replica |
| 7 | Ensure the Replica has caught up | trepctl -host host1 status | Replica | Primary | Replica |
| 8 | Switch Primary back to `host1` | See Section 7.11, "Switching Primary Hosts" | Primary | Replica | Replica |

## 7.13.3. Performing Maintenance on an Entire Dataservice

To perform maintenance on all of the machines within a replicator service, a rolling sequence of maintenance must be performed carefully on each machine in a structured way. In brief, the sequence is as follows

1. Perform maintenance on each of the current Replicas
2. Switch the Primary to one of the already maintained Replicas
3. Perform maintenance on the old Primary (now in Replica state)
4. Switch the old Primary back to be the Primary again

A more detailed sequence of steps, including the status of each datasource in the dataservice, and the commands to be performed, is shown in the table below. The table assumes a three-node dataservice (one Primary, two Replicas), but the same principles can be applied to any Primary/Replica dataservice:

| Step | Description | Command | host1 | host2 | host3 |
|---|---|---|---|---|---|
| 1 | Initial state | | Primary | Replica | Replica |
| 2 | Set the Replica `host2` offline | trepctl -host host2 offline | Primary | Offline | Replica |
| 3 | Perform maintenance | | Primary | Offline | Replica |
| 4 | Validate the `host2` server configuration | tpm validate | Primary | Offline | Replica |
| 5 | Set Replica `host2` online | trepctl -host host2 online | Primary | Replica | Replica |
| 6 | Ensure the Replica (`host2`) has caught up | trepctl -host host2 status | Primary | Replica | Replica |
| 7 | Set the Replica `host3` offline | trepctl -host host3 offline | Primary | Replica | Offline |
| 8 | Perform maintenance | | Primary | Replica | Offline |
| 9 | Validate the `host3` server configuration | tpm validate | Primary | Replica | Offline |
| 10 | Set the Replica `host3` online | trepctl -host host3 online | Primary | Replica | Replica |
| 11 | Ensure the Replica (`host3`) has caught up | trepctl -host host3 status | Primary | Replica | Replica |
| 12 | Switch Primary to `host2` | See Section 7.11, "Switching Primary Hosts" | Replica | Primary | Replica |
| 13 | Set the Replica `host1` | trepctl -host host1 offline | Offline | Primary | Replica |
| 14 | Perform maintenance | | Offline | Primary | Replica |
| 15 | Validate the `host1` server configuration | tpm validate | Offline | Primary | Replica |
| 16 | Set the Replica `host1` online | trepctl -host host3 online | Replica | Primary | Replica |

| Step | Description | Command | host1 | host2 | host3 |
|---|---|---|---|---|---|
| 17 | Ensure the Replica (`host1`) has caught up | trepctl -host host1 status | Primary | Replica | Replica |
| 18 | Switch Primary back to `host1` | See Section 7.11, "Switching Primary Hosts" | Primary | Replica | Replica |

## 7.13.4. Upgrading or Updating your JVM

When upgrading your JVM version or installation, care should be taken as changing the JVM will momentarily remove and replace required libraries and components which may upset the operation of Tungsten Cluster while the upgrade or update takes place.

For this reason, JVM updates or changes must be treated as an OS upgrade or event, requiring a Primary switch and controlled stopping of services during the update process.

A sample sequence for this in a 3-node cluster is described below:

| Step | Description | Command | host1 | host2 | host3 |
|---|---|---|---|---|---|
| 1 | Initial state | | Primary | Replica | Replica |
| 2 | Stop all services on `host2`. | stopall | Primary | Stopped | Replica |
| 3 | Update the JVM | | Primary | Stopped | Replica |
| 4 | Start all services on `host2` Replica. | startall | Primary | Replica | Replica |
| 5 | Stop all services on `host3`. | stopall | Primary | Replica | Stopped |
| 6 | Update the JVM | | Primary | Replica | Stopped |
| 7 | Start all services on `host3` Replica. | startall | Primary | Replica | Replica |
| 8 | Stop all services on `host1`. | stopall | Stopped | Replica | Replica |
| 9 | Update the JVM | | Stopped | Replica | Replica |
| 10 | Start all services on `host1` Primary. | startall | Primary | Replica | Replica |

The status of all services on all hosts should be checked to ensure they are running and operating as normal once the update has been completed.

# 7.14. Upgrading Tungsten Replicator

To upgrade an existing installation of Tungsten Replicator, the upgrade must be performed from a staging directory containing the new release. The process updates the Tungsten Replicator software and restarts the replicator service using the current configuration.

## 7.14.1. Upgrading Tungsten Replicator using tpm

To upgrade an existing installation, the new distribution must be downloaded and unpacked, and the included tpm command used to update the installation. The upgrade process implies a small period of downtime for the replicator as the updated versions of the tools are restarted, but downtime is deliberately kept to a minimum, and the replicator should be in the same operational state once the upgrade has finished as it was when the upgrade was started.

### Warning

Before performing an upgrade, please ensure that you have checked the Appendix B, *Prerequisites* and the appropriate Release Notes for the version you are upgrading, as software and system requirements may have changed between versions and releases.

The method for the upgrade process depends on whether you installed via the ini method or via the staging method

Upgrading an ini based Installation

1. On each host in your deployment, download the release package and place this in your staging directory, typically `/opt/continuent/software`.
2. Unpack the release package:

```
shell> tar zxf tungsten-replicator-7.1.2-81.tar.gz
```

3. Change to the unpackaged directory:

```
shell> cd tungsten-replicator-7.1.2-81
```

4. Run the validation process:

```
shell> ./tools/tpm validate-update
```

> **Note**
>
> The validate process will check that pre-requisities are in place and that tpm can safely upgrade the software. Any errors that are reported should be handled before proceding

5. Run the upgrade process:

```
shell> ./tools/tpm update --replace-release
```

The update process should now be complete. The current version can be confirmed by using trepctl status.

Upgrading a staging based installation

1. Download the release package and place this in the staging directory on your staging host, typically `/opt/continuent/software`.

   If you are unsure which host and directory this should be, execute the following on any host:

```
shell> tpm query staging
```

   The output of this command will display the host and directory

2. Unpack the release package:

```
shell> tar zxf tungsten-replicator-7.1.2-81.tar.gz
```

3. Change to the unpackaged directory:

```
shell> cd tungsten-replicator-7.1.2-81
```

4. Fetch a copy of the existing configuration information:

```
shell> ./tools/tpm fetch --hosts=host1,host2,autodetect --user=tungsten --directory=/opt/continuent
```

> **Important**
>
> You must use the version of tpm from within the staging directory (./tools/tpm) of the new release, not the tpm installed with the current release.

   The fetch command to tpm supports the following arguments:

   - `--hosts` [409]

     A comma-separated list of the known hosts in the deployment. If `autodetect` is included, then tpm will attempt to determine other hosts in the deployment by checking the configuration files for host values.

   - `--user` [429]

     The username to be used when logging in to other hosts.

   - `--directory` [404]

     The installation directory of the current Tungsten Replicator installation. If `autodetect` is specified, then tpm will look for the installation directory by checking any running Tungsten Replicator processes.

   The current configuration information will be retrieved to be used for the upgrade:

```
shell> ./tools/tpm fetch --hosts=host1,host2 --directory=/opt/continuent --user=tungsten
..
NOTE  >> Configuration loaded from host1,host2
```

5. Run the validation process:

```
shell> ./tools/tpm validate-update
```

> **Note**
>
> The validate process will check that pre-requisities are in place and that tpm can safely upgrade the software on all hosts. Any errors that are reported should be handled before proceding

6. Run the upgrade process:

```
shell> ./tools/tpm update --replace-release
```

The update process should now be complete. The current version can be confirmed by using trepctl status.

## 7.14.2. Installing an Upgraded JAR Patch

### Warning

> The following instructions should only be used if Continuent Support have explicitly provided you with a customer JAR file designed to address a problem with your deployment.

If a custom JAR has been provided by Continuent Support, the following instructions can be used to install the JAR into your installation.

1. Determine your staging directory or untarred installation directory:

```
shell> tpm query staging
```

Go to the appropriate host (if necessary) and the staging directory.

```
shell>  cd tungsten-replicator-7.1.2-81
```

2. Change to the correct directory:

```
shell> cd tungsten-replicator/lib
```

3. Copy the existing JAR to a backup file:

```
shell> cp tungsten-replicator.jar tungsten-replicator.jar.orig
```

4. Copy the replacement JAR into the directory:

```
shell> cp /tmp/tungsten-replicator.jar .
```

5. Change back to the root directory of the staging directory:

```
shell> cd ../..
```

6. Update the release:

```
shell> ./tools/tpm update --replace-release
```

## 7.14.3. Installing Patches

### Warning

> This procedure should only be followed with the advice and guidance of a Continuent Support Engineer.

There are two ways we can patch the running environment, and the method chosen will depend on the severity of the patch and whether or not your use case would allow for a maintenance window

- Upgrade using a full software update following the standard upgrade procedures
- Use the patch command to patch just the files necessary

From time to time, Continuent may provide you with a patch to apply as a quicker way to fix small issues. Patched software will always be provided in a subsequent release so the manual patch method described here should only be used as a temporary measure to patch a live installation when a full software update may not immediately be possible

You will have been supplied with a file containing the patch, for the purpose of this example we will assume the file you have been given is called `undeployallnostop.patch`

1. On each node of your installation:

   a. Copy the supplied patch file to the host

   b. From the installed directory (Typically this would be `/opt/continuent`) issue the following:

```
shell> cd /opt/continuent/tungsten
```

```
shell> patch -p1 -i undeployallnostop.patch
```

#### Warning

> If a tpm update --replace-release is issued from the original software staging directory, the manual patch applied above will be over-written and removed.
>
> The manual patch method is a temporary approach to patching a running environment, but is not a total replacement for a proper upgrade.
>
> Following a manual patch, you MUST plan to upgrade the staged software to avoid reverting to an unpatched system.
>
> If in doubt, always check with a Continuent Support Engineer.

## 7.14.4. Upgrading to v7.0.0+

#### Warning

> v7 is a major release with many changes, specifically to security. At this time, upgrading directly to v7 is only supported from v5 onwards. If security is NOT enabled in your installation, then upgrading from an older release may work, however any issues encountered will not be addressed and upgrading to v6 first will be the advised route.

#### Warning

> Whilst every care has been taken to ensure upgrades are as smooth and easy as possible, ALWAYS ensure full backups are taken before proceeding, and if possible, test the upgrade on a non-Production environment first.

### 7.14.4.1. Background

#### 7.14.4.1.1. v6 (and earlier) behavior

Prior to v7, Tungsten came with security turned OFF through the tpm flag `disable-security-controls` [404] set to `true` by default. This flag, when set to `false` would translate to the following settings being applied:

```
file-protection-level=0027 [408]
rmi-ssl=true [407]
thl-ssl=true [408]
rmi-authentication=true [407]
jgroups-ssl=true [407]
```

This would enable SSL communication between Tungsten components. However, connection to the database remained unencrypted, which would translate to the following settings being applied:

```
datasource-enable-ssl=false [400]
connector-ssl=false
```

Setting these to true is possible, however there are many more manual steps that would have been required.

#### 7.14.4.1.2. New behavior in v7

v7 enables full security by default, so the `disable-security-controls` [404] flag will default to `false` when not specified.

In addition to the default value changing, `disable-security-controls` [404] now enables encrypted communication to the database. Setting this value to `false`, now translates to the following settings being applied:

```
file-protection-level=0027 [408]
rmi-ssl=true [407]
thl-ssl=true [408]
rmi-authentication=true [407]
jgroups-ssl=true [407]
datasource-enable-ssl=true [400]
connector-ssl=true
```

#### 7.14.4.1.3. Summary

In summary, this change in behavior means that upgrades need to be handled with care and appropriate decisions being made, both by the tpm process, and by the "human" to decide on what end result is desired. The various options and examples are outlined in the following sections of this document.

## 7.14.4.2. Upgrade Decisions

### 7.14.4.2.1. Keep existing level of security

This is the easiest and smoothest approach. tpm will process your configuration and do its best to maintain the same level of security. In order to achieve that, tpm will dynamically update your configuration (either the `tungsten.ini` file for INI installs, or the `deploy.cfg` for staging installs) with additional properties to adjust the level of security to match.

The properties that tpm will add to your configuration will be some or all of the following depending on the initial starting point of your configuration:

```
disable-security-controls [404]
connector-rest-api-ssl
manager-rest-api-ssl
replicator-rest-api-ssl [421]
datasource-enable-ssl [400]
enable-connector-ssl
```

You can now proceed with the upgrade, refer to Section 7.14.4.6, "Steps to upgrade using tpm" for the required steps

### 7.14.4.2.2. Apply new recommendations and setup security

The following security setting levels can be enabled, and will require user action prior to upgrading. These are:

1. Internal Encryption and Authentication
2. Tungsten to Database Encryption
3. API SSL

Applying all of the above steps will bring full security, equivalent to the default v7 configuration.

The steps to enable will depend on what (if any) security is enabled in your existing installation. The following sections outline the steps required to be performed to enable security for each of the various layers. To understand whether you have configured any of the various layers of security, the following summary will help to understand your configuration:

No Security

If no security has been configured, the installation that you are starting from will have `disable-security-controls=true` [404] (or it will not supplied at all) and no additional securoty properties will be supplied.

Partial Security

The installation that you are starting from will have partial security in place. This could be a combination of any of the following:

- Internal encryption is configured (`disable-security-controls=false` [404]), and/or
- Replicator to the database encryption is enabled (`datasource-enable-ssl=true` [400] or `repl-datasource-enable-ssl=true` [400])

To upgrade and enable security, you should follow one or more of the following steps based on your requirements. At a minimum, the first step should always be included, the remaining steps are optional.

1. Section 7.14.4.3, "Setup internal encryption and authentication"
2. Section 7.14.4.4, "Enable Tungsten to Database Encryption" (if required)

## 7.14.4.3. Setup internal encryption and authentication

Prior to running the upgrade, you need to manually create the keystore, to do this follow these steps on one host, and then copy the files to all other hosts in your topology:

```
db1> mkdir /etc/tungsten/secure
db1> keytool -genseckey -alias jgroups -validity 3650 -keyalg Blowfish -keysize 56 \
-keystore /etc/tungsten/secure/jgroups.jceks -storepass tungsten -keypass tungsten -storetype JCEKS
```

If you have an INI based install, and this is the only level of security you plan on configuring you should now copy these new keystores to all other hosts in your topology. If you plan to enable SSL at the other remaining layers, or you use a Staging based install, then skip this copy step.

```
db1> for host in db2 db3 db4 db5 db6; do
ssh ${host} mkdir /etc/tungsten/secure
scp /etc/tungsten/secure/*.jceks ${host}:/etc/tungsten/secure
done
```

Enabling internal encryption and authentication will also enable API SSL by default.

If you need to enable encryption to the underlying database, now proceed to the next step Section 7.14.4.4, "Enable Tungsten to Database Encryption" before running the upgrade, otherwise you can then start the upgrade by following the steps in Section 7.14.4.6, "Steps to upgrade using tpm".

The following additional configuration properties will need adding to your existing configuration. The suggested process based on an INI or Staging based install are outlined in the final upgrade steps referenced above.

```
disable-security-controls=false [404]
replicator-rest-api-ssl=true [421]
java-jgroups-keystore-path=/etc/tungsten/secure/jgroups.jceks [410]
```

## 7.14.4.4. Enable Tungsten to Database Encryption

The following prerequisite steps must be performed before continuing with this step

- Section 7.14.4.3, "Setup internal encryption and authentication"
- Section 7.14.4.5, "Enable MySQL SSL"

In this step, you pre-create the various keystores required and register the MySQL certificates for Tungsten. Execute all of the following steps on a single host, for example, db1. In the example below it is assumed that the mysql certificates reside in `/etc/mysql/certs`. If you use the example syntax below, you will also need to ensure the following directory exists: `/etc/tungsten/secure`

These commands will import the MySQL certificates into the required Tungsten truststores.

```
db1> keytool -importkeystore -srckeystore /etc/mysql/certs/client-cert.p12 -srcstoretype PKCS12 \
-destkeystore /etc/tungsten/secure/keystore.jks -deststorepass tungsten -srcstorepass tungsten

db1> keytool -import -alias mysql -file /etc/mysql/certs/ca.pem -keystore /etc/tungsten/secure/truststore.ts \
-storepass tungsten -noprompt
```

If you have an INI based install, you should now copy all of the generated files to all other hosts in your topology. If you use a Staging based install, then skip this copy step.

```
db1> for host in db2 db3 db4 db5 db6; do
ssh ${host} mkdir /etc/tungsten/secure
scp /etc/tungsten/secure/*.jceks ${host}:/etc/tungsten/secure
scp /etc/tungsten/secure/*.jks ${host}:/etc/tungsten/secure
scp /etc/tungsten/secure/*.ts ${host}:/etc/tungsten/secure
done
```

Once the steps above have been performed, you can then continue with the upgrade, following the steps outlined in Section 7.14.4.6, "Steps to upgrade using tpm"

The following additional configuration properties will need adding to your existing configuration. The suggested process based on an INI or Staging based install are outlined in the final upgrade steps referenced above.

```
datasource-enable-ssl=true [400]
java-truststore-path=/etc/tungsten/secure/truststore.ts [411]
java-truststore-password=tungsten [411]
java-keystore-path=/etc/tungsten/secure/keystore.jks [410]
java-keystore-password=tungsten [410]
datasource-mysql-ssl-cert=/etc/mysql/certs/client-cert.pem [401]
datasource-mysql-ssl-key=/etc/mysql/certs/client-key.pem [401]
datasource-mysql-ssl-ca=/etc/mysql/certs/ca.pem [401]
```

## 7.14.4.5. Enable MySQL SSL

A prerequisite to enabling full security, is to enable SSL within your database if this isn't already configured. To do this, we can use the mysql_ssl_rsa_setup tool supplied with most distributions of MySQL. If you do not have this tool, or require more detail, you can refer to Section 6.10.1, "Enabling Database SSL". The steps below summarise the process using the mysql_ssl_rsa_setup

1. The first step is to setup the directories for the certs, perform this on ALL hosts in your topology:

   ```
   shell> sudo mkdir -p /etc/mysql/certs
   shell> sudo chown -R tungsten: /etc/mysql/certs/
   ```

   NB: The ownership is temporarily set to tungsten so that the subsequent scp will work between hosts.

2. This next step should be performed on just one single host, for the purpose of this example we will use db1 as the host:

   ```
   db1> mysql_ssl_rsa_setup -d /etc/mysql/certs/
   db1> openssl pkcs12 -export -inkey /etc/mysql/certs/client-key.pem \
   -name mysql -in /etc/mysql/certs/client-cert.pem -out /etc/mysql/certs/client-cert.p12 \
   ```

```
-passout pass:tungsten
```

> **Important**
>
> When using OpenSSL 3.0 with Java 1.8, you MUST add the `-legacy` option to the openssl command.

```
db1> for host in db2 db3 db4 db5 db6; do
scp /etc/mysql/certs/* ${host}:/etc/mysql/certs
done
```

3. Next, on every host we need to reset the directory ownership

```
shell> sudo chown -R mysql: /etc/mysql/certs/
shell> sudo chmod g+r /etc/mysql/certs/client-*
```

4. Now on every host, we need to reconfigure MySQL. Add the following properties into your `my.cnf`

```
[mysqld]
ssl-ca=/etc/mysql/certs/ca.pem
ssl-cert=/etc/mysql/certs/server-cert.pem
ssl-key=/etc/mysql/certs/server-key.pem

[client]
ssl-cert=/etc/mysql/certs/client-cert.pem
ssl-key=/etc/mysql/certs/client-key.pem
ssl-ca=/etc/mysql/certs/ca.pem
```

5. Restart MySQL for the new settings to take effect

```
shell> sudo service mysqld restart
```

## 7.14.4.6. Steps to upgrade using tpm

When you are ready to perform the upgrade, the following steps should be followed:

### 7.14.4.6.1. Steps for INI Based Installations

1. If no additional steps taken, and you wish to maintain the same level of security, skip Step 2, and proceed directly to Step 3.
2. Update your `tungsten.ini` and include some, or all, of the options below depending on which steps you took earlier. All entries should be placed within the `[defaults]` stanza.

```
disable-security-controls=false [404]
replicator-rest-api-ssl=true [421]
java-jgroups-keystore-path=/etc/tungsten/secure/jgroups.jceks [410]
```

If "Tungsten to Database Encryption" IS configured, also add:

```
datasource-enable-ssl=true [400]
java-truststore-path=/etc/tungsten/secure/truststore.ts [411]
java-truststore-password=tungsten [411]
java-keystore-path=/etc/tungsten/secure/keystore.jks [410]
java-keystore-password=tungsten [410]
datasource-mysql-ssl-cert=/etc/mysql/certs/client-cert.pem [401]
datasource-mysql-ssl-key=/etc/mysql/certs/client-key.pem [401]
datasource-mysql-ssl-ca=/etc/mysql/certs/ca.pem [401]
```

If "Tungsten to Database Encryption" IS NOT configured, also add:

```
datasource-enable-ssl=false [400]
```

> **Important**
>
> If `start-and-report=true` [424], remove this value or set to `false`

3. Obtain the TAR or RPM package for your installation. If using a TAR file unpack this into your software staging tree, typically `/opt/continuent/software`. If you use the INI install method, this needs to be performed on every host. For staging install, this applies to the staging host only.
4. Change into the directory for the software

```
shell> cd /opt/continuent/software/tungsten-replicator-7.1.2-81
```

5. Issue the following command on all hosts.

```
shell> tools/tpm update --replace-release
```

6. Finally, you will need to sync the new certificates, created by the upgrade, to all hosts This step will be required even if you have disabled security as these files will be used by the API and also, if you choose to enable it, THL Encryption.

   From one host, copy the certificate and keystore files to ALL other hosts in your topology. The following scp command is an example assuming you are issuing from db1, and the install directory is /opt/continuent:

```
db1> for host in db2 db3 db4 db5 db6; do
scp /opt/continuent/share/[jpt]* ${host}:/opt/continuent/share
scp /opt/continuent/share/.[jpt]* ${host}:/opt/continuent/share
done
```

   **Note**

   The examples assume you have the ability to scp between hosts as the tungsten OS user. If your security restrictions do not permit this, you will need to use alternative procedures appropriate to your environment to ensure these files are in sync across all hosts before continuing.

   If the files are not in sync between hosts, the software will fail to start!

7. Restart all tungsten components, one host at a time

```
shell> replicator restart
```

### 7.14.4.6.2. Steps for Staging Based Installations

1. Obtain the TAR or RPM package for your installation. If using a TAR file unpack this into your software staging tree, typically `/opt/continuent/software`. If you use the INI install method, this needs to be performed on every host. For staging install, this applies to the staging host only.

2. Change into the directory for the software and fetch the configuration, e.g

```
shell> cd /opt/continuent/software/tungsten-replicator-7.1.2-81
shell> tpm reverse > deploy.sh
```

3. If no additional steps taken, and you wish to maintain the same level of security, skip Step 4, and proceed directly to Step 5.

4. Edit the deploy.sh file just created, and include some, or all, of the options below depending on which steps you took earlier (They should be placed within the `defaults`.

```
--disable-security-controls=false [404]
--replicator-rest-api-ssl=true [421]
--java-jgroups-keystore-path=/etc/tungsten/secure/jgroups.jceks [410]
```

   If "Tungsten to Database Encryption" IS configured, also add:

```
--datasource-enable-ssl=true [400]
--java-truststore-path=/etc/tungsten/secure/truststore.ts [411]
--java-truststore-password=tungsten [411]
--java-keystore-path=/etc/tungsten/secure/keystore.jks [410]
--java-keystore-password=tungsten [410]
--datasource-mysql-ssl-cert=/etc/mysql/certs/client-cert.pem [401]
--datasource-mysql-ssl-key=/etc/mysql/certs/client-key.pem [401]
--datasource-mysql-ssl-ca=/etc/mysql/certs/ca.pem [401]
```

   If "Tungsten to Database Encryption" IS NOT configured, also add:

```
--datasource-enable-ssl=false [400]
```

   **Important**

   If `start-and-report=true` [424], remove this value or set to `false`

   An example of a BEFORE and AFTER edit including all options:

```
shell> cat deploy.sh
# BEFORE
tools/tpm configure defaults \
--reset \
--application-port=3306 \
--disable-security-controls=true \
--install-directory=/opt/continuent \
--mysql-allow-intensive-checks=true \
--profile-script=/home/tungsten/.bash_profile \
--replication-password=secret \
--replication-user=tungsten \
--start-and-report=true \
```

```
--user=tungsten
# Options for the nyc data service
tools/tpm configure nyc \
--master=db1 \
--slaves=db2 \
--topology=master-slave
```

```
shell> cat deploy.sh
# BEFORE
tools/tpm configure defaults \
--reset \
--application-password=secret \
--application-port=3306 \
--application-user=app_user \
--install-directory=/opt/continuent \
--mysql-allow-intensive-checks=true \
--profile-script=/home/tungsten/.bash_profile \
--replication-password=secret \
--replication-user=tungsten \
--user=tungsten \
--disable-security-controls=false \
--replicator-rest-api-ssl=true \
--datasource-enable-ssl=true \
--java-jgroups-keystore-path=/etc/tungsten/secure/jgroups.jceks \
--java-truststore-path=/etc/tungsten/secure/truststore.ts \
--java-truststore-password=tungsten \
--java-keystore-path=/etc/tungsten/secure/keystore.jks \
--java-keystore-password=tungsten \

# Options for the nyc data service
# Options for the nyc data service
tools/tpm configure nyc \
--master=db1 \
--slaves=db2 \
--topology=master-slave
```

5. Next, source the file to load the configuration and then execute the update:

```
shell> source deploy.sh
shell> tools/tpm update --replace-release
```

6. Finally, you will need to sync the new certificates, created by the upgrade, to all hosts This step will be required even if you have disabled security as these files will be used by the API and also, if you choose to enable it, THL Encryption.

   From one host, copy the certificate and keystore files to ALL other hosts in your topology. The following scp command is an example assuming you are issuing from db1, and the install directory is /opt/continuent:

```
db1> for host in db2 db3 db4 db5 db6; do
scp /opt/continuent/share/[jpt]* ${host}:/opt/continuent/share
scp /opt/continuent/share/.[jpt]* ${host}:/opt/continuent/share
done
```

   **Note**

   The examples assume you have the ability to scp between hosts as the tungsten OS user. If your security restrictions do not permit this, you will need to use alternative procedures appropriate to your environment to ensure these files are in sync across all hosts before continuing.

   If the files are not in sync between hosts, the software will fail to start!

7. Restart all tungsten components, one host at a time

```
shell> replicator restart
```

## 7.14.4.7. Optional Post-Upgrade steps to configure API

Once the upgrade has been completed, if you plan on using the API you will need to complete a few extra steps before you can use it. By default, after installation the API will only allow the ping method and the createAdminUser method.

To open up the API and access all of its features, you will need to configure the API User. To do this, execute the following on all hosts {Setting the value of pass to your preferred password}:

```
shell> curl -k -H 'Content-type: application/json' --request POST 'https://127.0.0.1:8096/api/v2/createAdminUser?i-am-sure=true' \
> --data-raw '{
>   "payloadType": "credentials",
>   "user":"tungsten",
>   "pass":"security"
> }'
```

For more information on using the new API, please refer to Chapter 10, *Tungsten REST API (APIv2)*

# 7.15. Monitoring Tungsten Cluster

It is your responsibility to properly monitor your deployments of Tungsten Cluster and Tungsten Replicator. The minimum level of monitoring must be done at three levels. Additional monitors may be run depending on your environment but these three are required in order to ensure availability and uptime.

1. Make sure the appropriate Tungsten Cluster and Tungsten Replicator services are running.
2. Make sure all datasources and replication services are `ONLINE` [195].
3. Make sure replication latency is within an acceptable range.

> **Important**
>
> Special consideration must be taken if you have multiple installations on a single server. That applies for clustering and replication or multiple replicators.
>
> These three points must be checked for all directories where Tungsten Cluster or Tungsten Replicator are installed. In addition, all servers should be monitored for basic health of the processors, disk and network. Proper alerting and graphing will prevent many issues that will cause system failures.

## 7.15.1. Managing Log Files with logrotate

You can manage the logs generated by Tungsten Cluster using logrotate.

- `trepsvc.log`

```
/opt/continuent/tungsten/tungsten-replicator/log/trepsvc.log {
        notifempty
        daily
        rotate 3
        missingok
        compress
        copytruncate
}
```

## 7.15.2. Monitoring Status Using cacti

Graphing Tungsten Replicator data is supported through Cacti extensions. These provide information gathering for the following data points:

- Applied Latency
- Sequence Number (Events applied)
- Status (Online, Offline, Error, or Other)

To configure the Cacti services:

1. Download both files from https://github.com/continuent/monitoring/tree/master/cacti
2. Place the PHP script into `/usr/share/cacti/scripts`.
3. Modify the installed PHP file with the appropriate `$ssh_user` and `$tungsten_home` location from your installation:
   - `$ssh_user` should match the `user` used during installation.
   - `$tungsten_home` is the installation directory and the `tungsten` subdirectory. For example, if you have installed into `/opt/continuent`, use `/opt/continuent/tungsten`.

   Add SSH arguments to specify the correct `id_rsa` file if needed.
4. Ensure that the configured `$ssh_user` has the correct SSH authorized keys to login to the server or servers being monitored. The user must also have the correct permissions and rights to write to the cache directory.
5. Test the script by running it by hand:

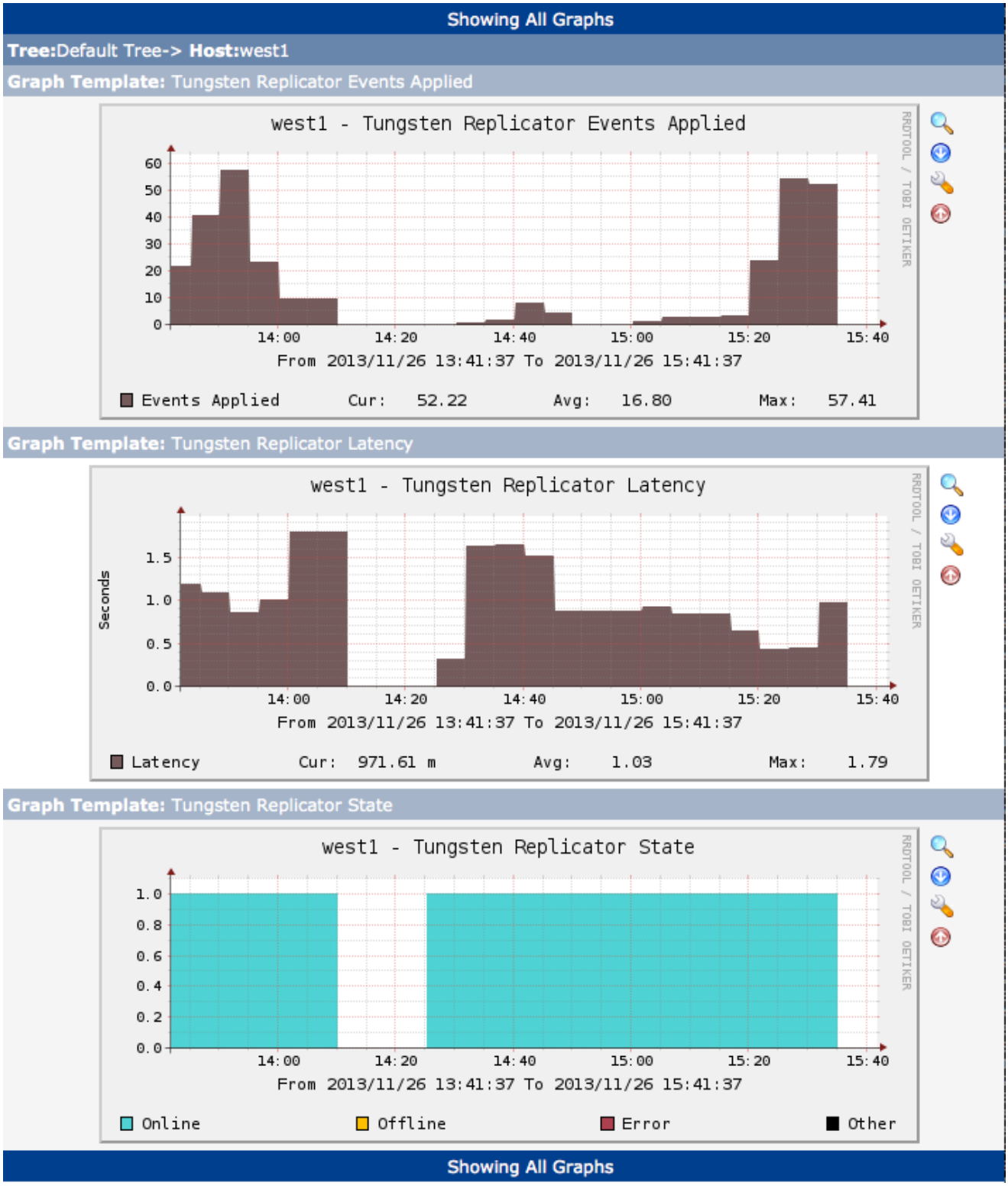
   ```
   shell> php -q /usr/share/cacti/scripts/get_replicator_stats.php --hostname replserver
   ```

   If you are using multiple replication services, add `--service servicename` to the command.
6. Import the XML file as a Cacti template.

7. Add the desired graphs to your servers running Tungsten Replicator. If you are using multiple replications services, you'll need to specify the desired service to graph. A graph must be added for each individual replication service.

Once configured, graphs can be used to display the activity and availability.

Figure 7.1. Cacti Monitoring: Example Graphs



## 7.15.3. Monitoring Status Using nagios

Integration with Nagios is supported through a number of scripts that output information in a format compatible with the Nagios NRPE plugin. Using the plugin the check commands, such as check_tungsten_latency can be executed and the output parsed for status information.

The available commands are:

- check_tungsten_latency
- check_tungsten_online
- check_tungsten_services

To configure the scripts to be executed through NRPE:

1. Install the Nagios NRPE server.
2. Start the NRPE daemon:

```
shell> sudo /etc/init.d/nagios-nrpe-server start
```

3. Add the IP of your Nagios server to the `/etc/nagios/nrpe.cfg` configuration file. For example:

```
allowed_hosts=127.0.0.1,192.168.2.20
```

4. Add the Tungsten check commands that you want to execute to the `/etc/nagios/nrpe.cfg` configuration file. For example:

```
command[check_tungsten_online]=/opt/continuent/tungsten/cluster-home/bin/check_tungsten_online
```

5. Restart the NRPE service:

```
shell> sudo /etc/init.d/nagios-nrpe-server start
```

6. If the commands need to be executed with superuser privileges, the `/etc/sudo` or `/etc/sudoers` file must be updated to enable the commands to be executed as root through sudo as the `nagios` user. This can be achieved by updating the configuration file, usually performed by using the visudo command:

```
nagios          ALL=(tungsten)  NOPASSWD: /opt/continuent/tungsten/cluster-home/bin/check*
```

   In addition, the sudo command should be added to the Tungsten check commands within the Nagios `nrpe.cfg`, for example:

```
command[check_tungsten_online]=/usr/bin/sudo -u tungsten /opt/continuent/tungsten/cluster-home/bin/check_tungsten_online
```

   Restart the NRPE service for these changes to take effect.

7. Add an entry to your Nagios `services.cfg` file for each service you want to monitor:

```
define service {
        host_name database
        service_description     check_tungsten_online
        check_command           check_nrpe! -H $HOSTADDRESS$  -t 30 -c check_tungsten_online
        retry_check_interval    1
        check_period            24x7
        max_check_attempts      3
        flap_detection_enabled  1
        notifications_enabled   1
        notification_period     24x7
        notification_interval   60
        notification_options    c,f,r,u,w
        normal_check_interval   5
}
```

The same process can be repeated for all the hosts within your environment where there is a Tungsten service installed.

## 7.15.4. Monitoring Status Using Prometheus Exporters

Continuent has introduced basic support for using Prometheus and Grafana to monitor Tungsten nodes. As of Tungsten software v6.1.4, five key Prometheus exporters have been added to the distribution.

### 7.15.4.1. Monitoring Status with Exporters Overview

The exporters will allow a Prometheus server to gather metrics for:

- The underlying node "hardware" using an external `node_exporter` binary added to the distribution.

- The running MySQL server using an external `mysqld_exporter` binary added to the distribution.
- The Tungsten Replicator using new built-in functionality in the replicator binary.

A new command, tmonitor, has been included to assist with the management and testing of the exporters.

To manually test any exporter, get the URL `http://{theHost}:{thePort}/metrics` either via the curl command or a browser.

```
shell> curl -s 'http://localhost:9100/metrics' | wc -l
916

shell> curl -s 'http://localhost:9100/metrics' | head -10
# HELP go_gc_duration_seconds A summary of the GC invocation durations.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 1.0129e-05
go_gc_duration_seconds{quantile="0.25"} 1.3347e-05
go_gc_duration_seconds{quantile="0.5"} 1.8895e-05
go_gc_duration_seconds{quantile="0.75"} 3.095e-05
go_gc_duration_seconds{quantile="1"} 0.00104028
go_gc_duration_seconds_sum 10.43582891
go_gc_duration_seconds_count 173258
# HELP go_goroutines Number of goroutines that currently exist.
```

The below table describes the exporters and the ports they listen on.

| Exporter | Port | Description | Scope |
|---|---|---|---|
| node | 9100 | Metrics for the underlying node "hardware" | External |
| mysql | 9104 | Metrics for the MySQL server | External |
| replicator | 8091 | Metrics for the Tungsten Replicator | Internal |
| manager | 8092 | Metrics for the Tungsten Manager | Internal |
| connector | 8093 | Metrics for the Tungsten Connector | Internal |

## 7.15.4.2. Customizing the Prometheus Exporter Configuration

If the default ports (`8091`, `8092` & `8093`) are in conflict, it is possible to change them.

Add one line per component to `/etc/tungsten/tungsten.ini` under the `[defaults]` section:

```
shell> vi /etc/tungsten/tungsten.ini
[defaults]
property=replicator.prometheus.exporter.port=28091
property=manager.prometheus.exporter.port=28092
property=connector.prometheus.exporter.port=28093
…
```

### Warning

Either SHUN the individual nodes one at a time and WELCOME each one after running the update, or set policy to MAINTENANCE via cctrl and update all nodes, then set the policy back to AUTOMATIC when all nodes have been completed.

Inform the running processes of the changed configuration:

```
shell> tpm update
```

> **Important**
>
> You may need to restart the Manager, Replicator or Connector, depending on what was changed.

Verify that the port is listening:

```
shell> sudo netstat -pan | grep 28091
```

### 7.15.4.3. Disabling the Prometheus Exporters

Add the following to `/etc/tungsten/tungsten.ini` under the `[defaults]` section:

The Prometheus exporters are all enabled by default. It is simple to disable them using the following procedure.

Add one line per component to `/etc/tungsten/tungsten.ini` under the `[defaults]` section:

```
shell> vi /etc/tungsten/tungsten.ini
[defaults]
property=manager.prometheus.exporter.enabled=false
property=replicator.prometheus.exporter.enabled=false
property=connector.prometheus.exporter.enabled=false
…
```

> **Warning**
>
> Either SHUN the individual nodes one at a time and WELCOME each one after running the update, or set policy to MAINTENANCE via cctrl and update all nodes, then set the policy back to AUTOMATIC when all nodes have been completed.

Inform the running processes of the changed configuration:

```
shell> tpm update
```

> **Important**
>
> You may need to restart the Manager, Replicator or Connector, depending on what was changed.

Verify that the port is no longer listening:

```
shell> sudo netstat -pan | grep 8091
```

### 7.15.4.4. Managing and Testing Exporters Using the tmonitor Command

tmonitor is a simple tool for the management and testing of Prometheus exporters.

Exporters that require an external binary to function (i.e. node_exporter and mysqld_exporter) are considered to have an External Scope.

Exporters that do not require an external binary to function (i.e. manager, replicator and connector) are considered to have an Internal Scope.

By default, tmonitor {action} will act upon all available exporters.

If any exporter is specified on the CLI, then only those listed on the CLI will be acted upon.

The curl command must be available in the PATH for the `status` and `test` actions to function.

tmonitor status will use curl to test the exporter on `localhost`.

tmonitor test will use curl to fetch and print the metrics from one or more exporters on `localhost`.

tmonitor install will configure the specified exporter (or all external exporters if none is specified) to start at boot.

tmonitor remove will stop the specified exporter (or all external exporters if none is specified) from starting at boot.

Both the `install` and `remove` actions will attempt to auto-detect the boot sub-system. Currently, `init.d` and `systemd` are supported.

```
shell> tmonitor help
...
>>> Usage:
```

```
   tmonitor [args] {action}

   = Actions available for all exporter scopes:

      status - validate the service via curl (short output)
      test - validate the service via curl (full output)

   = Actions available for External-scope exporters only:

      start - launch the exporter process
      stop - kill the exporter process

      install - configure the exporter to start at boot
      remove - stop the exporter from starting at boot


>>> Arguments:

   [-h|--help]
   [-v|--verbose]
   [--force] Required for certain MySQL-specific operations

   [-f|--filter {string}] Limit the `tmonitor test` output based on this string match
   [-t|--tungsten] Set the `tmonitor test` filter to 'tungsten_'

   [-i|--internal] Only act upon exporters with an internal scope
   [-e|-x|--external] Only act upon exporters with an external scope

   --internal and --external may not be specified together.

   = Internal Scope Exporters:

     [-C|--connector] Specify the Tungsten Connector exporter
     [-M|--manager] Specify the Tungsten Manager exporter
     [-R--replicator] Specify the Tungsten Replicator exporter

   = External Scope Exporters:

     [-m|--mysql|--mysqld] Specify the MySQL exporter
     [-n|--node] Specify the Node exporter
```

Example: View the status of all exporters:

```
shell> tmonitor status
Tungsten Connector exporter running ok on port 8093
Tungsten Manager exporter running ok on port 8092
MySQL exporter running ok on port 9104
Node exporter running ok on port 9100
Tungsten Replicator exporter running ok on port 8091
All 5 exporters are running ok (Up: Tungsten Connector, Tungsten Manager, MySQL, Node, Tungsten Replicator)
```

Example: Start all exporters:

```
shell> tmonitor start
tungsten@db1-demo:/home/tungsten # tmonitor start
Node exporter started successfully on port 9100.
MySQL exporter started successfully on port 9104.

shell> tmonitor start
The Node exporter is already running
The MySQL exporter is already running
```

Example: Test all exporters:

```
shell> tmonitor test | wc -l
3097

shell> tmonitor test | grep '== '
== Metrics for the connector exporter:
== Metrics for the manager exporter:
== Metrics for the mysql exporter:
== Metrics for the node exporter:
== Metrics for the replicator exporter:

shell> tmonitor test | less
```

Example: Stop all exporters:

```
shell> tmonitor stop
All exporters stopped
```

Example: View the status of all exporters filtered by scope:

```
shell> tmonitor status -i
Tungsten Connector exporter running ok on port 8093
Tungsten Manager exporter running ok on port 8092
Tungsten Replicator exporter running ok on port 8091
All 3 internal exporters are running ok (Up: Tungsten Connector, Tungsten Manager, Tungsten Replicator)

shell> tmonitor status -x
MySQL exporter running ok on port 9104
Node exporter running ok on port 9100
All 2 external exporters are running ok (Up: MySQL, Node)
```

Example: Install init.d boot scripts for all external exporters:

```
shell> tmonitor install
ERROR: You must be root to install or remove boot services.

Please be sure to run tmonitor as root via sudo, for example:

sudo /opt/continuent/tungsten/cluster-home/bin/tmonitor install


shell> sudo /opt/continuent/tungsten/cluster-home/bin/tmonitor install
node_exporter init.d boot script installed and activated

Use either `sudo service node_exporter start` or `tmonitor --node start` now to start the Node exporter.

mysqld_exporter init.d boot script installed and activated

Use either `sudo service mysqld_exporter start` or `tmonitor --mysql start` now to start the MySQL exporter.
```

Example: Install systemd boot scripts for all external exporters:

```
shell> tmonitor install
ERROR: You must be root to install or remove boot services.

Please be sure to run tmonitor as root via sudo, for example:

sudo /opt/continuent/tungsten/cluster-home/bin/tmonitor install


shell> sudo /opt/continuent/tungsten/cluster-home/bin/tmonitor install
Created symlink from /etc/systemd/system/multi-user.target.wants/node_exporter.service to /etc/systemd/system/node_exporter.service.
node_exporter systemd boot script installed and enabled

Use either `sudo systemctl start node_exporter` or `tmonitor --node start` now to start the Node exporter.

Created symlink from /etc/systemd/system/multi-user.target.wants/mysqld_exporter.service to /etc/systemd/system/mysqld_exporter.service.
mysqld_exporter systemd boot script installed and enabled

Use either `sudo systemctl start mysqld_exporter` or `tmonitor --mysql start` now to start the MySQL exporter.
```

Example: Remove init.d boot scripts for all external exporters:

```
shell> tmonitor remove
ERROR: You must be root to install or remove boot services.

Please be sure to run tmonitor as root via sudo, for example:

sudo /opt/continuent/tungsten/cluster-home/bin/tmonitor remove

shell> sudo /opt/continuent/tungsten/cluster-home/bin/tmonitor remove
node_exporter is still running, unable to remove. Please run either `tmonitor --node stop` or `sudo service node_exporter stop`, then retry this operation

shell> tmonitor stop
Node exporter stopped
MySQL exporter stopped

shell> sudo /opt/continuent/tungsten/cluster-home/bin/tmonitor remove
node_exporter init.d boot script de-activated and removed
mysqld_exporter init.d boot script de-activated and removed
```

Example: Remove systemd boot scripts for all external exporters:

```
shell> tmonitor stop
Node exporter stopped
MySQL exporter stopped

shell> tmonitor remove
ERROR: You must be root to install or remove boot services.

Please be sure to run tmonitor as root via sudo, for example:

sudo /opt/continuent/tungsten/cluster-home/bin/tmonitor remove
```

```
shell> sudo /opt/continuent/tungsten/cluster-home/bin/tmonitor remove
node_exporter systemd unit boot script disabled and removed
mysqld_exporter systemd unit boot script disabled and removed
```

The tmonitor command is located in the `$CONTINUENT_ROOT/tungsten/cluster-home/bin` directory.

> **Note**
>
> The tmonitor command will only be available in the `PATH` if the Tungsten software has been installed with the configuration option `profile-script` [417] included.

### 7.15.4.5. Monitoring Node Status Using the External node_exporter

The node_exporter will respond to requests on port `9100`, path `/metrics`

The tmonitor command is the best way to manage the node_exporter binary.

For example, to test a running node_exporter service:

```
shell> tmonitor --node test | wc -l
869

shell> tmonitor --node test | head -10
==========================================================================================
== Metrics for the node exporter:
==========================================================================================
# HELP go_gc_duration_seconds A summary of the GC invocation durations.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 1.0214e-05
go_gc_duration_seconds{quantile="0.25"} 1.3524e-05
go_gc_duration_seconds{quantile="0.5"} 2.2065e-05
go_gc_duration_seconds{quantile="0.75"} 4.1943e-05
go_gc_duration_seconds{quantile="1"} 0.003692845
```

To start the node_exporter binary (only), and then get the status:

```
shell> tmonitor start --node
shell> tmonitor status --node
Node exporter running ok
```

> **Warning**
>
> The node_exporter command is not included in the `PATH` unless you add it manually.

The node_exporter binary is located in the `$CONTINUENT_ROOT/tungsten/cluster-home/prometheus` directory.

### 7.15.4.6. Monitoring MySQL Server Status Using the External mysqld_exporter

The mysqld_exporter will respond to requests on port `9104`, path `/metrics`

The mysqld_exporter command will read MySQL server connection information from the `~/.my.cnf` file. Since the default listener port for the MySQL server is `13306`, the file must contain at least:

```
shell> cat ~/.my.cnf
[client]
port=13306
user={tungsten_database_user_here}
password={tungsten_database_password_here}
```

The tmonitor command is the best way to manage the mysqld_exporter binary.

For example, to test a running mysqld_exporter service:

```
shell> tmonitor --mysqld test | wc -l
1813

shell> tmonitor --mysqld test | head -10
==========================================================================================
== Metrics for the mysql exporter:
==========================================================================================
# HELP go_gc_duration_seconds A summary of the GC invocation durations.
 # TYPE go_gc_duration_seconds summary
 go_gc_duration_seconds{quantile="0"} 1.0051e-05
 go_gc_duration_seconds{quantile="0.25"} 1.4436e-05
 go_gc_duration_seconds{quantile="0.5"} 3.3925e-05
 go_gc_duration_seconds{quantile="0.75"} 6.3136e-05
```

```
go_gc_duration_seconds{quantile="1"} 0.000551545
```

To start the mysqld_exporter binary (only), and then get the status:

```
shell> tmonitor start --mysqld
shell> tmonitor status --mysqld
mysqld exporter running ok
```

#### Warning

The mysqld_exporter command is not included in the `PATH` unless you add it manually.

The mysqld_exporter binary is located in the `$CONTINUENT_ROOT/tungsten/cluster-home/prometheus` directory.

### 7.15.4.7. Monitoring Tungsten Replicator Status Using the Built-In Exporter

The replicator will respond to requests on port `8093`, path `/metrics`

The tmonitor command is the best way to test the replicator exporter.

For example, to test a running replicator exporter service:

```
shell> tmonitor --replicator test | wc -l
148

shell> tmonitor -t -R test
==========================================================================================
== Metrics for the replicator exporter:
==========================================================================================
# HELP tungsten_replicator_version The Tungsten Clustering software version number
# TYPE tungsten_replicator_version gauge
tungsten_replicator_version{version="Tungsten Clustering 7.0.0 build 478",vendor="Continuent",name="Tungsten Replicator",} 1.0
# HELP tungsten_replicator_service Replicator service value
# TYPE tungsten_replicator_service gauge
tungsten_replicator_service{service="east",role="master",state="online",} 1.0
tungsten_replicator_service{service="east_from_west",role="relay",state="online",} 1.0
# HELP tungsten_replicator_seqno Replicator min/max/current sequence number value
# TYPE tungsten_replicator_seqno gauge
tungsten_replicator_seqno{service="east",seqno="minimum",} 0.0
tungsten_replicator_seqno{service="east",seqno="maximum",} 741693.0
tungsten_replicator_seqno{service="east",seqno="current",} 741693.0
tungsten_replicator_seqno{service="east_from_west",seqno="minimum",} 0.0
tungsten_replicator_seqno{service="east_from_west",seqno="maximum",} 638682.0
tungsten_replicator_seqno{service="east_from_west",seqno="current",} 638682.0
# HELP tungsten_replicator_latency Replicator applied/relative latency value
# TYPE tungsten_replicator_latency gauge
tungsten_replicator_latency{service="east",latency="applied",} 0.754
tungsten_replicator_latency{service="east",latency="relative",} 0.762
tungsten_replicator_latency{service="east_from_west",latency="applied",} 0.738
tungsten_replicator_latency{service="east_from_west",latency="relative",} 0.763
```

## 7.16. Rebuilding THL on the Primary

If THL is lost on a Primary before the events contained within it have been applied to the Replica(s), the THL will need to be rebuilt from the existing MySQL binary logs.

#### Important

If the MySQL binary logs no longer exist, then recovery of the lost transactions in THL will NOT be possible.

The basic sequence of operation for recovering the THL on both Primary and Replicas is:

1. Gather the failing requested sequence numbers from all Replicas:

```
shell> trepctl status

pendingError           : Event extraction failed
pendingErrorCode       : NONE
pendingErrorEventId    : NONE
pendingErrorSeqno      : -1
pendingExceptionMessage: Client handshake failure: Client response validation failed:
Master log does not contain requested transaction:
master source ID=db1 client source ID=db2 requested seqno=4 client epoch number=0 master min seqno=8 master max seqno=8
```

   In the above example, when Replica db2 comes back online, it requests a copy of the last seqno in local thl (4) from the Primary db1 to compare for data integrity purposes, which the Primary no longer has.

Keep a note of the lowest sequence number and the host that it is on across all Replicas for use in the next step.

2. On the Replica with the lowest failing requested seqno, get the epoch, source-id and event-id (binlog position) from the THL using the command thl list -seqno [268] specifying the sequence number above. This information will be needed on the extractor (Primary) in a later step. For example:

```
tungsten@db2:/opt/replicator> thl list -seqno 4

SEQ# = 4 / FRAG# = 0 (last frag)
- TIME = 2017-07-14 14:49:00.0
- EPOCH# = 0
- EVENTID = mysql-bin.000009:0000000000001844;56
- SOURCEID = db1
- METADATA = [mysql_server_id=33155307;dbms_type=mysql;tz_aware=true;is_metadata=true; »
              service=east;shard=#UNKNOWN;heartbeat=NONE]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [##charset = UTF-8, autocommit = 1, sql_auto_is_null = 0,
             foreign_key_checks = 1, unique_checks = 1, time_zone = '+00:00',
             sql_mode = 'NO_ENGINE_SUBSTITUTION,STRICT_TRANS_TABLES,IGNORE_SPACE',
             character_set_client = 33, collation_connection = 33, collation_server = 8]
- SCHEMA = tungsten_east
- SQL(0) = UPDATE tungsten_east.heartbeat SET source_tstamp= '2017-07-14 14:49:00', $raquo;
           salt= 5, name= 'NONE'  WHERE id= 1
```

There are two more ways of getting the same information using the dsctl command, so use the one you are most comfortable with:

```
tungsten@db2:/opt/replicator> dsctl get
[{"extract_timestamp":"2017-07-14 14:49:00.0","eventid":"mysql-bin.000009:0000000000001844;56",»
  "fragno":0,"last_frag":true,"seqno":4,"update_timestamp":"2017-07-14 14:49:00.0",»
  "shard_id":"#UNKNOWN","applied_latency":0,"epoch_number":0,"task_id":0,"source_id":"db1"}]
```

```
tungsten@db2:/opt/replicator> dsctl get -ascmd
dsctl set -seqno 4 -epoch 0 -event-id "mysql-bin.000009:0000000000001844;56;566" -source-id "db1"
```

3. Clear all THL on the Primary since it is no longer needed by any Replicas:

```
shell> thl purge
```

4. Use the dsctl command on the Primary with the values we got from the Replica with the lowest seqno to tell the Primary replicator to begin generating THL starting from that event in the MySQL binary logs:

   Note: If you used the dsctl get -ascmd earlier, you may use that provided command now, just add the `-reset` argument at the end.

```
shell> dsctl set -seqno 4 -epoch 0 -event-id "mysql-bin.000009:0000000000001844;56;566" -source-id "db1" -reset
```

5. Switch the Primary to online state:

```
shell> trepctl online
```

6. Switch the Replicas to online state once the Primary is fully online:

```
shell> trepctl online
```

# 7.17. THL Encryption and Compression

The ability to Compress and/or Encrypt THL was introduced in v7.0.0

Encryption is applied to THL on disk, in flight encryption is handled by enabling the various SSL features of the Replicator

Compression can be enabled in-flight by changing the various configuration properties, and Compression on disk can be enabled/disabled either dynamically or by changing the various configuration properties.

The following sections explain enabling/disabling these features in more detail.

## 7.17.1. In-Flight Compression

Compression occurs "in-flight" and is requested by the client replicator prior to fetching the THL from the remote THL Server.

*Enabling THL Compression*

The following property should be added to your configuration:

For ini installations:

```
repl-thl-client-serialization=[SERIALIZATION PROTOCOL]
```

For example:

```
repl-thl-client-serialization=PROTOBUF
```

For Staging installations:

```
--repl-thl-client-serialization=[SERIALIZATION PROTOCOL]
```

For example:

```
--repl-thl-client-serialization=DEFLATE
```

Valid values for the protocol are: `LEGACY, JAVA, PROTOBUF` or `DEFLATE`

The default for this property if not supplied is `LEGACY`. This retains the behavior in versions prior to v7.0.0 and has the same effect as disabling compression.

`DEFLATE` offers the highest level of compression, but at the cost of being slower during the compression and decompression stages

`JAVA` should not be used in production, and is mainly used for testing purposes.

The THL Server can be configured to accept one or more of these protocols. By default, a server will support ALL protocols. This can be adjusted as follows:

```
repl-thl-server-serialization=[COMMA SEPARATED LIST OF PROTOCOLS]
```

For example, the following would disable `DEFLATE`:

```
repl-thl-server-serialization=LEGACY,PROTOBUF
```

If a client asks for a protocol that is not enabled, it will fall back to `LEGACY`

## 7.17.2. Encryption and Compression On-Disk

THL Encryption and Compression On-Disk can be enabled at install time or dynamically.

The following property should be added to your configuration:

For ini installations:

```
replicator-store-thl-encrypted=true|false
replicator-store-thl-compressed=true|false
```

For Staging installations:

```
--replicator-store-thl-encrypted=true|false
--replicator-store-thl-compressed=true|false
```

By default, both encryption and compression are disabled.

To change these settings dynamically, the service will need to be put offline first. This will force the replicator to rotate to a new THL log file which will use the new settings.

> **Note**
>
> If only enabled/disabled dynamicaly, the settings WILL persist on a replicator restart.

The commands to enable or disable these settings are:

```
shell> trepctl [-service servicename] thl -compression {enable|disable}
shell> trepctl [-service servicename] thl -encryption {enable|disable}
```

The full steps to enable encryption follows:

```
shell> trepctl [-service servicename] offline
shell> trepctl [-service servicename] thl -encryption enable
shell> trepctl [-service servicename] online
```

As a result, after this command, thl index command will show the newly generated THL log file as encrypted:

```
shell> thl [-service servicename] index
….
LogIndexEntry thl.data.0000000008(42:42)
```

```
LogIndexEntry thl.data.0000000009(43:43) - ENCRYPTED (thl.ct1691.16)
```

Encryption uses dedicated keystore and truststore (named by default `tungsten_thl_keystore.jks` and `tungsten_thl_truststore.ts`). Losing these files will make encrypted THL log files impossible to be decoded.

The Replicator can generate new keys to be used. These keys will then be sent through the THL protocol to other nodes.

To generate a new key , the following command needs to be executed while the service is online:

```
shell> trepctl [-service servicename] thl -encryption genkey
```

This will result in a new THL log file to be started, using the new generated key, as shown by thl index command:

```
shell> thl [-service servicename] index
….
LogIndexEntry thl.data.0000000008(42:42)
LogIndexEntry thl.data.0000000009(43:45) - ENCRYPTED (thl.ct1691.16)
LogIndexEntry thl.data.0000000010(46:46) - ENCRYPTED (thl.ct1691.46)
```

# Chapter 8. Command-line Tools

Tungsten Replicator is supplied with a number of different command-line tools and utilities that help to install manage, control and provide additional functionality on top of the core Tungsten Replicator product.

The content in this chapter provides reference information for using and working with all of these tools. Usage and operation with these tools in particular circumstances and scenarios are provided in other chapters. For example, deployments are handled in Chapter 2, *Deployment Overview*, although all deployments rely on the tpm command.

Commands related to the deployment

- tpm — Tungsten package manager
- ddlscan — Data definition layer scanner and translator

Commands related to the core Tungsten Replicator

- trepctl — replicator control
- multi_trepctl — multi-replicator control
- thl — examine Tungsten History Log contents

Commands related to managing Tungsten Replicator deployments

- tprovision — provision or reprovision a Replica from an existing Primary or Replica database
- tungsten_read_master_events — read Primary events to determine the correct log position

Commands related to the Hadoop Deployments

- load-reduce-check — build DDL, materialize and compare replicated data
- materialize — materializer of views of replicated data into tables

Commands related to monitoring

- tmonitor — management and testing of external Prometheus exporters
- tungsten_monitor — provides a mechanism for monitoring the replicator state
- tungsten_send_diag — assists with diag and file uploads to Continuent support

## 8.1. The clean_release_directory Command

The clean_release_directory is located in the `tools` directory removes older releases of the installed product from the installation directory. Over time, as tpm update the configuration or new releases of the product, new directories with the full release information are created, but old ones are not removed in case you need to go back to a previous release.

The clean_release_directory command removes all but the five most recent installs and the current release. For example, with the following directory:

```
shell> ls -l /opt/continuent/releases
drwxrwxr-x 17 mc mc 4096 Jul  7 15:36 ./
drwxr-xr-x  9 mc mc 4096 Jul  7 15:36 ../
drwxrwxr-x  2 mc mc 4096 Jul  7 15:36 install/
drwxr-xr-x  5 mc mc 4096 Jul  7 14:35 tungsten-replicator-5.2.0-218_pid16197/
drwxr-xr-x  5 mc mc 4096 Jul  7 15:36 tungsten-replicator-5.2.0-219_pid10303/
drwxr-xr-x  5 mc mc 4096 Jul  7 15:35 tungsten-replicator-5.2.0-219_pid1393/
drwxr-xr-x  5 mc mc 4096 Jul  7 15:34 tungsten-replicator-5.2.0-219_pid23112/
drwxr-xr-x  5 mc mc 4096 Jul  7 15:35 tungsten-replicator-5.2.0-219_pid24935/
drwxr-xr-x  5 mc mc 4096 Jul  7 15:35 tungsten-replicator-5.2.0-219_pid26720/
drwxr-xr-x  5 mc mc 4096 Jul  7 15:35 tungsten-replicator-5.2.0-219_pid28491/
drwxr-xr-x  5 mc mc 4096 Jul  7 15:35 tungsten-replicator-5.2.0-219_pid30270/
drwxr-xr-x  5 mc mc 4096 Jul  7 15:35 tungsten-replicator-5.2.0-219_pid32041/
drwxr-xr-x  5 mc mc 4096 Jul  7 15:35 tungsten-replicator-5.2.0-219_pid3212/
drwxr-xr-x  5 mc mc 4096 Jul  7 15:35 tungsten-replicator-5.2.0-219_pid4983/
drwxr-xr-x  5 mc mc 4096 Jul  7 15:35 tungsten-replicator-5.2.0-219_pid6754/
drwxr-xr-x  5 mc mc 4096 Jul  7 15:36 tungsten-replicator-5.2.0-219_pid8530/
drwxrwxr-x  5 mc mc 4096 Jul  6 11:09 tungsten-replicator-5.2.0_pid2869/
```

### Warning

The clean_release_directory command removes old releases. Although this does not affect THL, stored data, or your configuration, it may remove working, but old, configurations, releases and versions of Tungsten Cluster.

Running clean_release_directory:

```
shell> ./tools/clean_release_directory
Deleting release directories in /opt/continuent; keeping the last five and current installation
Cleaning old releases from /opt/continuent
Deleting /opt/continuent/releases/tungsten-replicator-5.2.0-219_pid32041
Deleting /opt/continuent/releases/tungsten-replicator-5.2.0-219_pid30270
Deleting /opt/continuent/releases/tungsten-replicator-5.2.0-219_pid28491
Deleting /opt/continuent/releases/tungsten-replicator-5.2.0-219_pid26720
Deleting /opt/continuent/releases/tungsten-replicator-5.2.0-219_pid24935
Deleting /opt/continuent/releases/tungsten-replicator-5.2.0-219_pid23112
Deleting /opt/continuent/releases/tungsten-replicator-5.2.0-218_pid16197
Deleting /opt/continuent/releases/tungsten-replicator-5.2.0_pid2869
```

The resulting releases directory now contains a simpler list:

```
shell> /opt/continuent/releases/
total 36
drwxrwxr-x 9 mc mc 4096 Jul  7 15:52 ./
drwxr-xr-x 9 mc mc 4096 Jul  7 15:36 ../
drwxrwxr-x 2 mc mc 4096 Jul  7 15:36 install/
drwxr-xr-x 5 mc mc 4096 Jul  7 15:36 tungsten-replicator-5.2.0-219_pid10303/
drwxr-xr-x 5 mc mc 4096 Jul  7 15:35 tungsten-replicator-5.2.0-219_pid1393/
drwxr-xr-x 5 mc mc 4096 Jul  7 15:35 tungsten-replicator-5.2.0-219_pid3212/
drwxr-xr-x 5 mc mc 4096 Jul  7 15:35 tungsten-replicator-5.2.0-219_pid4983/
drwxr-xr-x 5 mc mc 4096 Jul  7 15:35 tungsten-replicator-5.2.0-219_pid6754/
drwxr-xr-x 5 mc mc 4096 Jul  7 15:36 tungsten-replicator-5.2.0-219_pid8530/
```

# 8.2. The check_tungsten_latency Command

The check_tungsten_latency command reports warning or critical status information depending on whether the latency across the nodes in the cluster is above a specific level.

Table 8.1. check_tungsten_latency Options

| Option | Description |
|---|---|
| `-c` | Report a critical status if the latency is above this level |
| `--perslave-perfdata` | Show the latency performance information on a per-Replica basis |
| `--perfdata` | Show the latency performance information |
| `-w` | Report a warning status if the latency is above this level |

The command outputs information in the following format:

```
LEVEL: DETAIL
```

Where DETAIL includes detailed information about the status report, and LEVEL is:

- CRITICAL — latency on at least one node is above the specified threshold level for a critical report. The host reporting the high latency will be included in the DETAIL portion:

  For example:

  ```
  CRITICAL: host2=0.506s
  ```

- WARNING — latency on at least one node is above the specified threshold level for a warning report. The host reporting the high latency will be included in the DETAIL portion:

  For example:

  ```
  WARNING: host2=0.506s
  ```

- OK — status is OK; the highest reported latency will be included in the output.

  For example:

  ```
  OK: All slaves are running normally (max_latency=0.506)
  ```

The `-w` and `-c` options must be specified on the command line, and the critical figure must be higher than the warning figure. For example:

```
shell> check_tungsten_latency -w 0.1 -c 0.5
CRITICAL: host2=0.506s
```

Performance information can be included in the output to monitor the status. The format for the output is included in the DETAIL block and separates the maximum latency information for each node with a semicolon, and the detail block with a pipe symbol. For example:

```
shell> check_tungsten_latency -w 1 -c 1 --perfdata
OK: All slaves are running normally (max_latency=0.506) |  max_latency=0.506;1;1;;
```

Performance information for all the Replicas in the cluster can be output by using the `--perslave-perfdata` option which must be used in conjunction with the `--perfdata` option:

```
shell> check_tungsten_latency -w 0.2 -c 0.5 --perfdata --perslave-perfdata
CRITICAL: host2=0.506s | host1=0.0;0.2;0.5;; host2=0.506;0.2;0.5;;
```

# 8.3. The check_tungsten_online Command

The check_tungsten_online command checks whether all the services for a given service and host are online and running.

### Table 8.2. check_tungsten_online Options

| Option | Description |
|---|---|
| `-h` | Display the help text |
| `-port` | RMI port for the replicator being checked |

By default, the script will check all manager and replication services for the localhost

The command outputs information in the following format:

```
LEVEL: DETAIL
```

Where DETAIL includes detailed information about the status report, and LEVEL is:

- CRITICAL — status is critical and requires immediate attention. This indicates that more than one service is not running.

  For example:

  ```
  CRITICAL: Replicator is not running
  ```

- WARNING — status requires attention. This indicates that one service within the system is not online.
- OK — status is OK.

  For example:

  ```
  OK: All services are online
  ```

This output is easily parseable by various monitoring tools, including Nagios NRPE, and can be used to monitor the status of your services quickly without resorting to using the full trepctl output.

For example:

```
shell> check_tungsten_online
OK: All services are online
```

# 8.4. The check_tungsten_services Command

The check_tungsten_services command provides a simple check to confirm whether configured services are currently running. The command must be executed with a command-line option specifying which services should be checked and confirmed.

### Table 8.3. check_tungsten_services Options

| Option | Description |
|---|---|
| `-h` | Display the help text. |
| `-r` | Check the replication services status. |

The command outputs information in the following format:

```
LEVEL: DETAIL
```

Where DETAIL includes detailed information about the status report, and LEVEL is:

- CRITICAL — status is critical and requires immediate attention.

For example:

```
CRITICAL: Replicator is not running
```

- OK — status is OK.

  For example:

```
OK: All services (Replicator) are online
```

This output is easily parseable by various monitoring tools, including Nagios NRPE, and can be used to monitor the status of your services quickly without restoring to using the full trepctl output.

> **Note**
>
> The check_tungsten_services only confirms that the services and processes are running; their state is not confirmed. To check state with a similar interface, use the check_tungsten_online command.

To check the services:

- To check the replicator services:

```
shell> check_tungsten_services -r
OK: All services (Replicator) are online
```

## 8.5. The deployall Command

The deployall tool installs the required startup scripts into the correct location so that all required services can be automatically started and stopped during the startup and shutdown of your server.

To use, the tool should be executed with superuser privileges, either directly using sudo, or by logging in as the superuser and running the command directly.

The script will automatically detect the initialization system in use (`systemd` or `initd`) and prefer `systemd` when both are available.

```
shell> sudo deployall
```

> **Important**
>
> In order for the configuration to persist during future updates, and/or to execute the deployall script at install time, you should add the `install=true` [409] tpm option to your configuration

The startup scripts are added to the correct run levels to enable operation during standard startup and shutdown levels.

> **Note**
>
> For systemd configurations only:
>
> For continuity of service reasons, the deployall script will NOT restart individual components if they had already been previously started by other methods. It will only install systemd scripts. This implies that, right after a call to deployall and before host/component restart, the system will stay in a mixed mode where systemd scripts are in place but components that were started without systemd, won't be controllable by it.
>
> In order to align the configuration, you will need to run
>
> ```
> shell> component stop sysd
> shell> sudo systemctl start tcomponent
> ```
>
> For example:
>
> ```
> shell> connector stop sysd
> shell> sudo systemctl start tconnector
> ```

> **Important**
>
> This note affects all versions up to and including v7.0.2. The workaround mentioned below will be included as a fix in the next patch release.
>
> When a service is controlled by systemd, the relevant OS limits (such as open file limits) are not controlled in the normal way (via settings in the limits.conf file) and therefore for systems with heavy workloads there is a risk that you may experience open file limits being exceeded which will affect the replicators stability.

> To resolve this, you must ensure you increase the limits for each service. Follow the steps below to do this:
>
> - Edit the service files in /etc/systemd/system. There will be one file for each service, for example `treplicator.service`
> - Under the `[service]` stanza, add the following:
>
> ```
> LimitNOFILE=65535
> ```
>
> - When you have changed all the files, reload the systemctl by issuing the following:
>
> ```
> systemctl daemon-reload
> ```
>
> - Retstart each of the services, e.g:
>
> ```
> systemctl restart treplicator
> ```
>
> Note that the restart will cause a momentary outage to each component therefore only do this when you are sure it is safe to do so.

See Section 2.5, "Configuring Startup on Boot".

To remove the scripts from the system, use undeployall.

# 8.6. The ddlscan Command

The ddlscan command scans the existing schema for a database or table and then generates a schema or file in a target database environment. For example, ddlscan is used in MySQL to Oracle heterogeneous deployments to translate the schema definitions within MySQL to the Oracle format. For more information on heterogeneous deployments, see Section 2.8, "Understanding Heterogeneous Deployments".

For example, to generate Oracle DDL from an existing MySQL database:

```
shell> ddlscan -user tungsten -url 'jdbc:mysql:thin://tr-source:13306/test' -pass password \
    -template ddl-mysql-oracle.vm -db test

SQL generated on Thu Sep 11 15:39:06 BST 2019 by ./ddlscan utility of Tungsten

url = jdbc:mysql:thin://tr-source:13306/test
user = tungsten
dbName = test
*/

DROP TABLE test.sales;
CREATE TABLE test.sales
(
  id NUMBER(10, 0) NOT NULL,
  salesman CHAR,
  planet CHAR,
  value FLOAT,
  PRIMARY KEY (id)
);
```

The format of the command is:

`ddlscan` [ `-conf path` ] [ `-db db` ] [ `-opt opt val` ] [ `-out file` ] [ `-pass secret` ] [ `-path path` ] [ `-rename file` ] [ `-service name` ] [ `-tableFile file` ] [ `-tables regex` ] [ `-template file` ] [ `-url jdbcUrl` ] [ `-user user` ]

The available options are as follows:

Table 8.4. ddlscan Command-line Options

| Option | Description |
|---|---|
| `-conf path` [243] | Path to a static-{svc}.properties file to read JDBC connection address and credentials |
| `-db db` [243] | Database to use (will substitute ${DBNAME} in the URL, if needed) |
| `-opt opt val` [244] | Option(s) to pass to template, try: -opt help me |
| `-out file` [244] | Render to file (print to stdout if not specified) |
| `-pass secret` [243] | JDBC password |
| `-path path` [243] | Add additional search path for loading Velocity templates |
| `-rename file` [243] | Definitions file for renaming schemas, tables and columns |

| Option | Description |
|---|---|
| `-service name` [243] | Name of a replication service instead of path to config |
| `-tableFile file` [243] | New-line separated definitions file of tables to find |
| `-tables regex` [243] | Comma-separated list of tables to find |
| `-template file` [243] | Specify template file to render |
| `-url jdbcUrl` [243] | JDBC connection string (use single quotes to escape) |
| `-user user` [243] | JDBC username |

ddlscan supports three different methods for execution:

- Using an explicit JDBC URL, username and password:

```
shell> ddlscan -user tungsten -url 'jdbc:mysql:thin://tr-hadoop1:13306/test' -user user \
    -pass password ...
```

  This is useful when a deployment has not already been installed.

- By specifying an explicit configuration file:

```
shell> ddlscan -conf /opt/continuent/tungsten/tungsten-replicator/conf/static-alpha.properties ...
```

- When an existing deployment has been installed, by specifying one of the active services:

```
shell> ddlscan -service alpha ...
```

In addition, the following two options must be specified on the command-line:

- The template to be used (using the `-template` [243] option) for the DDL translation must be specified on the command-line. A list of the support templates and their operation are available in Table 8.5, "ddlscan Supported Templates".
- The `-db` [243] parameter, which defines the database or schema that should be scanned. All tables are translated unless an explicit list, regex, or table file has been specified.

For example, to translate MySQL DDL to Oracle for all tables within the schema `test` using the connection to MySQL defined in the service `alpha`:

```
shell> ddlscan -service alpha -template ddl-mysql-oracle.vm -db test
```

ddlscan provides a series of additional command-line options, and a full list of the available templates.

## 8.6.1. Optional Arguments

The following arguments are optional:

- `-tables` [243]

  A comma-separate list of the tables to be extracted.

```
shell> ddlscan -service alpha -template ddl-mysql-oracle.vm -db test -tables typetwo,typethree
```

- `-tableFile` [243]

  A file containing a list of the files to be extracted. The file should be formatted as Comma Separated Values (CSV), only the first column is extracted. For example, the file:

```
typetwo,Table of type two customer forms
typethree,Table of type three customer forms
```

  Could be used with ddlscan:

```
shell> ddlscan -service alpha -template ddl-mysql-oracle.vm -db test -tableFile tablelist.txt
```

- `-rename` [243]

  A list of table renames which will be taken into account when generating target DDL. The format of the table matches the format of the `rename` filter.

- `-path` [243]

  The path to additional Velocity templates to be searched when specifying the template name.

- `-opt` [244]

  An additional option (and variable) which are supplied to be used within the template file. Different template files may support additional options for specifying alternative information, such as schema names, file locations and other values.

```
shell> ddlscan -service alpha -template ddl-mysql-oracle.vm -db test -opt schemaPrefix mysql_
```

- `-out` [244]

  Sends the generated DDL output to a file, in place of sending it to standard output.

- `-help` [244]

  Generates the help text of arguments.

## 8.6.2. Supported Templates and Usage

Table 8.5. ddlscan Supported Templates

| file | Description |
|---|---|
| `ddl-check-pkeys.vm` | Reports which tables are without primary key definitions |
| `ddl-mysql-hive-0.10.vm` | Generates DDL from a MySQL host suitable for the base tables in a Hadoop/Hive Environment |
| `ddl-mysql-hive-0.10-staging.vm` | Generates DDL from a MySQL host suitable for the staging tables in a Hadoop/Hive Environment |
| `ddl-mysql-hive-metadata.vm` | Generates metadata as JSON to be used within a Hadoop/Hive Environment |
| `ddl-mysql-oracle.vm` | Generates Oracle schema from a MySQL schema |
| `ddl-mysql-oracle-cdc.vm` | Generates Oracle tables with CDC capture information from a MySQL schema |
| `ddl-mysql-redshift.vm` | Generates DDL from a MySQL host suitable for the base tables in Amazon Redshift. |
| `ddl-mysql-redshift-staging.vm` | Generates DDL from a MySQL host suitable for the staging tables in Amazon Redshift. |
| `ddl-mysql-vertica.vm` | Generates DDL suitable for the base tables in HP Vertica |
| `ddl-mysql-vertica-staging.vm` | Generates DDL suitable for the staging tables in HP Vertica |
| `ddl-oracle-mysql.vm` | Generates DDL for MySQL tables from an Oracle schema |
| `ddl-oracle-mysql-pk-only.vm` | Generates Primary Key DDL statements from an Oracle database for MySQL |

### 8.6.2.1. `ddl-check-pkeys.vm`

The `ddl-check-pkeys.vm` template can be used to check whether specific tables within a schema do not have a primary key:

```
shell> ddlscan -template ddl-check-pkeys.vm \
 -user tungsten -pass password -db sales \
 -url jdbc:mysql://localhost:13306/sales
/*
SQL generated on Thu Sep 04 10:23:52 BST 2014 by ./ddlscan utility of Tungsten

url = jdbc:mysql://localhost:13306/sales
user = tungsten
dbName = sales
*/

/* ERROR: sales.dummy1 has no primary key! */*/
SQL generated on Thu Sep 04 10:23:52 BST 2014 by ./ddlscan utility of Tungsten

url = jdbc:mysql://localhost:13306/sales
user = tungsten
dbName = sales
*/

/* ERROR: sales.dummy2 has no primary key! */*/
SQL generated on Thu Sep 04 10:23:52 BST 2014 by ./ddlscan utility of Tungsten

url = jdbc:mysql://localhost:13306/sales
```

```
user = tungsten
dbName = sales
*/
```

For certain environments, particularly heterogeneous replication, the lack of primary keys can lead to inefficient replication, or even fail to replicate data at all.

### 8.6.2.2. ddl-mysql-hive-0.10.vm

Generates DDL suitable for a carbon-copy form of the table from the MySQL host:

```
shell> ddlscan -user tungsten -url 'jdbc:mysql://tr-hadoop1:13306/test' -pass password \
 -template ddl-mysql-hive-0.10.vm -db test
--
-- SQL generated on Thu Sep 11 12:57:11 BST 2014 by Tungsten ddlscan utility
--
-- url = jdbc:mysql://tr-hadoop1:13306/test
-- user = tungsten
-- dbName = test
--


DROP TABLE IF EXISTS test.sales;

CREATE TABLE test.sales
(
 id INT,
 salesman STRING,
 planet STRING,
 value DOUBLE )
;
```

Wherever possible, the closest Hive equivalent datatype is used for each source datatype, as follows:

| MySQL Datatype | Hive Datatype |
|---|---|
| DATETIME | STRING |
| TIMESTAMP | TIMESTAMP |
| DATE | STRING |
| YEAR | INT |
| TIME | STRING |
| TINYINT | TINYINT |
| TINYINT UNSIGNED | SMALLINT |
| SMALLINT | SMALLINT |
| SMALLINT UNSIGNED | INT |
| MEDIUMINT | INT |
| INT | INT |
| INT UNSIGNED | BIGINT |
| BIGINT | BIGINT |
| BIGINT UNSIGNED | STRING |
| DECIMAL | STRING |
| VARCHAR | STRING |
| CHAR | STRING |
| BINARY | BINARY |
| VARBINARY | BINARY |
| TEXT | STRING |
| BLOB | BINARY |
| FLOAT | DOUBLE |
| DOUBLE | DOUBLE |
| ENUM | STRING |
| SET | STRING |

| MySQL Datatype | Hive Datatype |
| --- | --- |
| BIT | STRING |

The template supports the following optional parameters to change behavior:

- -opt schemaPrefix

  A prefix to be placed in front of all schemas. For example, if called with *schemaPrefix* set to mysql_:

  ```
  shell> ddlscan ... -opt schemaPrefix mysql_
  ```

  The schema name will be prefixed, translating the schema name from sales into mysql_sales.

- -opt tablePrefix

  A prefix to be placed in front of all schemas. For example, if called with *tablePrefix* set to mysql_:

  ```
  shell> ddlscan ... -opt tablePrefix mysql_
  ```

  The table name will be prefixed, translating the tablename from sales into mysql_sales.

### 8.6.2.3. ddl-mysql-hive-0.10-staging.vm

Staging tables within Hive define the original table columns with additional columns to track the operation type, sequence number, timestamp and unique key for each row. For example, the table sales in MySQL:

```
mysql> describe sales;
+----------+----------+------+-----+---------+----------------+
| Field | Type | Null | Key | Default | Extra |
+----------+----------+------+-----+---------+----------------+
| id | int(11) | NO | PRI | NULL | auto_increment |
| salesman | char(20) | YES | | NULL | |
| planet | char(20) | YES | | NULL | |
| value | float | YES | | NULL | |
+----------+----------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

Generates the following Hive-compatible DDL when using this template:

```
shell> ddlscan -user tungsten -url 'jdbc:mysql://tr-hadoop1:13306/test' -pass password \
 -template ddl-mysql-hive-0.10-staging.vm -db test
--
-- SQL generated on Thu Sep 11 12:31:45 BST 2014 by Tungsten ddlscan utility
--
-- url = jdbc:mysql://tr-hadoop1:13306/test
-- user = tungsten
-- dbName = test
--


DROP TABLE IF EXISTS test.stage_xxx_sales;

CREATE EXTERNAL TABLE test.stage_xxx_sales
(
 tungsten_opcode STRING ,
 tungsten_seqno INT ,
 tungsten_row_id INT ,
 tungsten_commit_timestamp TIMESTAMP ,
 id INT,
 salesman STRING,
 planet STRING,
 value DOUBLE )
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\001' ESCAPED BY '\\'
LINES TERMINATED BY '\n'
STORED AS TEXTFILE LOCATION '/user/tungsten/staging/test/sales' ;
```

Wherever possible, the closest Hive equivalent datatype is used for each source datatype, see ddl-mysql-hive-0.10.vm for more information.

### 8.6.2.4. ddl-mysql-hive-metadata.vm

The Hadoop tools require information about the schema in JSON format so that the table names and primary key information can be used when materializing data from the staging tables into the base tables. This template generates that information in JSON format:

```
shell> ddlscan -user tungsten -url 'jdbc:mysql://tr-hadoop1:13306/test' -pass password \
 -template ddl-mysql-hive-metadata.vm -db test
```

```
{
 "tables": [
 {
 "schema": "test",
 "name": "sales",
 "keys": ["id"],
 "columns": [
 {"name": "id", "type": "INT"},
 {"name": "salesman", "type": "STRING"},
 {"name": "planet", "type": "STRING"},
 {"name": "value", "type": "DOUBLE"}
 ]
 }
 ]
}
```

### 8.6.2.5. `ddl-mysql-oracle.vm`

When translating MySQL tables to Oracle compatible schema, the following datatypes are migrated to their closest Oracle equivalent:

| MySQL Datatype | Oracle Datatype |
|---|---|
| `INT` | `NUMBER(10, 0)` |
| `BIGINT` | `NUMBER(19, 0)` |
| `TINYINT` | `NUMBER(3, 0)` |
| `SMALLINT` | `NUMBER(5, 0)` |
| `MEDIUMINT` | `NUMBER(7, 0)` |
| `DECIMAL(x,y)` | `NUMBER(x, y)` |
| `FLOAT` | `FLOAT` |
| `CHAR(n)` | `CHAR(n)` |
| `VARCHAR(n)` | `VARCHAR2(n) (n < 2000), CLOB n > 2000)` |
| `DATE` | `DATE` |
| `DATETIME` | `DATE` |
| `TIMESTAMP` | `DATE` |
| `TEXT` | `CLOB` |
| `BLOB` | `BLOB` |
| `ENUM(...)` | `VARCHAR(255)` |
| `ENUM(...)` | `VARCHAR(4000)` |
| `BIT(1)` | `NUMBER(1)` |

The following additional transformations happen automatically:

- Table names are translated to uppercase.
- Column names are translated to uppercase.
- If a column name is a reserved word in Oracle, then the column name has an underscore character appended (for example, `TABLE` becomes `TABLE_`).

In addition to the above translations, errors will be raised for the following conditions:

- If the table name starts with a number.
- If the table name exceeds 30 characters in length.
- If the table name is a reserved word in Oracle.

Warnings will be raised for the following conditions:

- If the column or column name started with a number.
- If the column name exceeds 30 characters in length, the column name will be truncated.
- If the column name is a reserved word in Oracle.

### 8.6.2.6. ddl-mysql-oracle-cdc.vm

The ddl-mysql-oracle-cdc.vm template generates identical tables in Oracle, from their MySQL equivalent, but with additional columns for CDC capture. For example:

```
shell> ddlscan -user tungsten -url 'jdbc:mysql://tr-hadoop1:13306/test' -pass password \
 -template ddl-mysql-oracle-cdc.vm -db test
/*
SQL generated on Thu Sep 11 13:17:05 BST 2014 by ./ddlscan utility of Tungsten

url = jdbc:mysql://tr-hadoop1:13306/test
user = tungsten
dbName = test
*/

DROP TABLE test.sales;
CREATE TABLE test.sales
(
 id NUMBER(10, 0) NOT NULL,
 salesman CHAR,
 planet CHAR,
 value FLOAT,
 CDC_OP_TYPE VARCHAR(1), /* CDC column */
 CDC_TIMESTAMP TIMESTAMP, /* CDC column */
 CDC_SEQUENCE_NUMBER NUMBER PRIMARY KEY /* CDC column */
);
```

For information on the datatypes translated, see ddl-mysql-oracle.vm.

### 8.6.2.7. ddl-mysql-redshift.vm

The ddl-mysql-redshift.vm template generates DDL for Amazon Redshift tables from MySQL schemas. For example:

```
CREATE TABLE test.all_mysql_types
(
 my_id INT,
 my_bit BOOLEAN /* BIT(1) */,
 my_tinyint SMALLINT /* TINYINT(4) */,
 my_boolean SMALLINT /* TINYINT(1) */,
 my_smallint SMALLINT,
 my_mediumint INT /* MEDIUMINT(9) */,
 my_int INT,
 my_bigint BIGINT,
 my_decimal_10_5 DECIMAL(10,5),
 my_float FLOAT,
 my_double DOUBLE PRECISION /* DOUBLE */,
 my_date DATE,
 my_datetime DATETIME,
 my_timestamp TIMESTAMP,
 my_time VARCHAR(17) /* WARN: no pure TIME type in Redshift */,
 my_year YEAR(4) /* ERROR: unrecognized (type=0, length=0) */,
 my_char_10 CHAR(10),
 my_varchar_10 VARCHAR(40) /* VARCHAR(10) */,
 my_tinytext VARCHAR(65535) /* WARN: MySQL TINYTEXT translated to max VARCHAR */,
 my_text VARCHAR(65535) /* WARN: MySQL TEXT translated to max VARCHAR */,
 my_mediumtext VARCHAR(65535) /* WARN: MySQL MEDIUMTEXT translated to max VARCHAR */,
 my_longtext VARCHAR(65535) /* WARN: MySQL LONGTEXT translated to max VARCHAR */,
 my_enum_abc VARCHAR(1) /* ENUM('A','B','C') */,
 my_set_def VARCHAR(65535) /* SET('D','E','F') */,
 PRIMARY KEY (my_id)
);
```

Columns are translated as follows:

| Oracle Datatype | Redshift Datatype |
|---|---|
| BIGINT | BIGINT |
| BINARY | BINARY, CHAR in 5.2.1 and later |
| BIT(1) | BOOLEAN |
| BIT | CHAR |
| BLOB | VARBINARY VARCHAR in 5.2.1 and later |
| CHAR | CHAR |
| DATE | DATE |
| DATETIME | DATETIME |

| Oracle Datatype | Redshift Datatype |
|---|---|
| DECIMAL | DECIMAL |
| DOUBLE | DOUBLE PRECISION |
| ENUM | VARCHAR |
| FLOAT | FLOAT |
| INT | INT |
| LONGBLOB | VARBINARY CHAR in 5.2.1 and later |
| LONGTEXT | VARCHAR |
| MEDIUMBLOB | VARBINARY CHAR in 5.2.1 and later |
| MEDIUMINT | INT |
| MEDIUMTEXT | VARCHAR |
| SET | VARCHAR |
| SMALLINT | SMALLINT |
| TEXT | VARCHAR |
| TIME | VARCHAR |
| TIMESTAMP | TIMESTAMP |
| TINYBLOB | VARBINARY CHAR in 5.2.1 and later |
| TINYINT | SMALLINT |
| TINYTEXT | VARCHAR |
| VARBINARY | VARBINARY CHAR in 5.2.1 and later |
| VARCHAR | VARCHAR |

In addition to these explicit changes, the following other considerations are taken into account:

- When translating the DDL for CHAR and VARCHAR columns, the actual column size is increased by a factor of four. This is because Redshift tables always stored data using 32-bit UTF characters and column sizes are in bytes, not characters. Therefore a CHAR(20) column is created as CHAR(80) within Redshift.
- TEXT columns are converted to a Redshift VARCHAR of 65535 in length (the maximum allowed).
- BLOB columns are converted to a Redshift VARBINARY of 65000 in length (the maximum allowed).
- BIT columns with a size of 1 are converted to Redshift BOOLEAN columns, larger sizes are converted to CHAR columns of 64 bytes in length.
- TIME columns are converted to a Redshift VARCHAR of 17 bytes in length since no explicit TIME type exists.

### 8.6.2.8. ddl-mysql-redshift-staging.vm

The ddl-mysql-redshift-staging.vm template generates DDL for Amazon Redshift tables from MySQL schemas. For example:

```
CREATE TABLE test.stage_xxx_all_mysql_types
(
 tungsten_opcode CHAR(2),
 tungsten_seqno INT,
 tungsten_row_id INT,
 tungsten_commit_timestamp TIMESTAMP,
 my_id INT,
 my_bit BOOLEAN /* BIT(1) */,
 my_tinyint SMALLINT /* TINYINT(4) */,
 my_boolean SMALLINT /* TINYINT(1) */,
 my_smallint SMALLINT,
 my_mediumint INT /* MEDIUMINT(9) */,
 my_int INT,
 my_bigint BIGINT,
 my_decimal_10_5 DECIMAL(10,5),
 my_float FLOAT,
 my_double DOUBLE PRECISION /* DOUBLE */,
 my_date DATE,
 my_datetime DATETIME,
 my_timestamp TIMESTAMP,
 my_time VARCHAR(17) /* WARN: no pure TIME type in Redshift */,
 my_year YEAR(4) /* ERROR: unrecognized (type=0, length=0) */,
 my_char_10 CHAR(10),
 my_varchar_10 VARCHAR(40) /* VARCHAR(10) */,
```

```
 my_tinytext VARCHAR(65535) /* WARN: MySQL TINYTEXT translated to max VARCHAR */,
 my_text VARCHAR(65535) /* WARN: MySQL TEXT translated to max VARCHAR */,
 my_mediumtext VARCHAR(65535) /* WARN: MySQL MEDIUMTEXT translated to max VARCHAR */,
 my_longtext VARCHAR(65535) /* WARN: MySQL LONGTEXT translated to max VARCHAR */,
 my_enum_abc VARCHAR(1) /* ENUM('A','B','C') */,
 my_set_def VARCHAR(65535) /* SET('D','E','F') */,
 PRIMARY KEY (tungsten_opcode, tungsten_seqno, tungsten_row_id)
);
```

The actual translation of datatypes is identical to that found in `ddl-mysql-redshift.vm`.

### 8.6.2.9. `ddl-mysql-vertica.vm`

The `ddl-mysql-vertica.vm` template generates DDL for generating tables within an HP Vertica database from an existing MySQL database schema. For example:

```
shell> ddlscan -user tungsten -url 'jdbc:mysql://tr-hadoop1:13306/test' -pass password \
 -template ddl-mysql-vertica.vm -db test
/*
SQL generated on Thu Sep 11 14:20:14 BST 2014 by ./ddlscan utility of Tungsten

url = jdbc:mysql://tr-hadoop1:13306/test
user = tungsten
dbName = test
*/
CREATE SCHEMA test;

DROP TABLE test.sales;

CREATE TABLE test.sales
(
 id INT ,
 salesman CHAR(20) ,
 planet CHAR(20) ,
 value FLOAT ) ORDER BY id;
```

Because Vertica does not explicitly support primary keys, a default projection for the key order is created based on the primary key of the source table.

The templates translates different datatypes as follows:

| MySQL Datatype | Vertica Datatype |
|---|---|
| DATETIME | DATETIME |
| TIMESTAMP | TIMESTAMP |
| DATE | DATE |
| TIME | TIME |
| TINYINT | TINYINT |
| SMALLINT | SMALLINT |
| MEDIUMINT | INT |
| INT | INT |
| BIGINT | INT |
| VARCHAR | VARCHAR |
| CHAR | CHAR |
| BINARY | BINARY |
| VARBINARY | VARBINARY |
| TEXT, TINYTEXT, MEDIUMTEXT, LONGTEXT | VARCHAR(65000) |
| BLOB, TINYBLOB, MEDIUMBLOB, LONGBLOB | VARBINARY(65000) |
| FLOAT | FLOAT |
| DOUBLE | DOUBLE PRECISION |
| ENUM | VARCHAR |
| SET | VARCHAR(4000) |
| BIT(1) | BOOLEAN |
| BIT | CHAR(64) |

In addition, the following considerations should be taken into account:

- `DECIMAL` MySQL type is not supported.
- `TEXT` types in MySQL are converted to a `VARCHAR` in Vertica of the maximum supported size.
- `BLOB` types in MySQL are converted to a `VARBINARY` in Vertica of the maximum supported size.
- `SET` types in MySQL are converted to a `VARCHAR` in Vertica of 4000 characters, designed to work in tandem with the `settostring` filter.
- `ENUM` types in MySQL are converted to a `VARCHAR` in Vertica of the size of the longest `ENUM` value, designed to work in tandem with the `enumtostring` filter.

### 8.6.2.10. `ddl-mysql-vertica-staging.vm`

The `ddl-mysql-vertica-staging.vm` template generates DDL for HP Vertica staging tables. These include the full table definition, in addition to three columns used to define the staging data, including the operation code, sequence number and unique row ID. For example:

```
shell> ddlscan -user tungsten -url 'jdbc:mysql://tr-hadoop1:13306/test' -pass password \
 -template ddl-mysql-vertica-staging.vm -db test
/*
SQL generated on Thu Sep 11 14:22:06 BST 2014 by ./ddlscan utility of Tungsten

url = jdbc:mysql://tr-hadoop1:13306/test
user = tungsten
dbName = test
*/
CREATE SCHEMA test;

DROP TABLE test.stage_xxx_sales;

CREATE TABLE test.stage_xxx_sales
(
 tungsten_opcode CHAR(1) ,
 tungsten_seqno INT ,
 tungsten_row_id INT ,
 id INT ,
 salesman CHAR(20) ,
 planet CHAR(20) ,
 value FLOAT ) ORDER BY tungsten_seqno, tungsten_row_id;
```

### 8.6.2.11. `ddl-oracle-mysql.vm`

The `ddl-oracle-mysql.vm` template generates the DDL required to create a schema within MySQL based on the existing Oracle schema. For example:

```
shell> ddlscan -service sales -template ddl-oracle-mysql.vm -db sales
/*
SQL generated on Thu Sep 11 04:29:08 PDT 2014 by ./ddlscan utility of Tungsten

url = jdbc:oracle:thin:@//tr-fromoracle1:1521/ORCL
user = SALES_PUB
dbName = sales
*/
/* ERROR: no tables found! Is database and tables option specified correctly? */

[tungsten@tr-fromoracle1 ~]$ ddlscan -service sales -template ddl-oracle-mysql.vm -db SALES
/*
SQL generated on Thu Sep 11 04:29:16 PDT 2014 by ./ddlscan utility of Tungsten

url = jdbc:oracle:thin:@//tr-fromoracle1:1521/ORCL
user = SALES_PUB
dbName = SALES
*/

DROP TABLE IF EXISTS sales.sample;
CREATE TABLE sales.sample
(
 id DECIMAL(38) /* NUMBER(38, ?) */ NOT NULL,
 msg CHAR(80),
 PRIMARY KEY (id)
) ENG
```

Columns are translated as follows:

| Oracle Datatype | MySQL Datatype |
|---|---|
| `DATE` | `DATETIME` |

| Oracle Datatype | MySQL Datatype |
|---|---|
| NUMBER(0) | NUMERIC |
| NUMBER(n) where n < 19 | INT |
| NUMBER(n) where n > 19 | BIGINT |
| NUMBER(n) where n < 3 | TINYINT |
| NUMBER(n) where n < 5 | SMALLINT |
| NUMBER(n) where n < 7 | MEDIUMINT |
| NUMBER(n) where n < 10 | INT |
| NUMBER(n) where n < 19 | BIGINT |
| NUMBER | DECIMAL |
| FLOAT | FLOAT |
| VARCHAR | VARCHAR |
| LONG | LONGTEXT |
| BFILE | VARCHAR(1024) |
| CHAR | CHAR |
| CLOB | LONGTEXT |
| BLOB | LONGBLOB |
| LONG RAW | LONGBLOB |
| TIMESTAMP | TIMESTAMP |
| RAW | VARBINARY |

The following additional transformations happen automatically:

- If a column name is a reserved word in MySQL, then the column name has an underscore character appended (for example, TABLE becomes TABLE_).

An error is raised in the following conditions:

- If the size of a FLOAT is larger than 53 points of precision.

### 8.6.2.12. ddl-oracle-mysql-pk-only.vm

The ddl-oracle-mysql-pk-only.vm template generates alter table statements to add the primary key, as determined from the Oracle primary key or index information. For example:

```
shell> ddlscan -service hadoop -template ddl-oracle-mysql-pk-only.vm -db HADOOP
/*
SQL generated on Thu Sep 11 06:17:28 PDT 2014 by ./ddlscan utility of Tungsten

url = jdbc:oracle:thin:@//tr-fromoracle1:1521/ORCL
user = HADOOP_PUB
dbName = HADOOP
*/

ALTER TABLE hadoop.sample ADD PRIMARY KEY (ID);
```

Note that it does not generate table DDL, only statements to alter existing tables with primary key information.

## 8.7. The dsctl Command

The dsctl command provides a simplified interface into controlling the datasource within a replication scenario to set the current replication position. Because dsctl uses the built-in datasource connectivity of the replicator, differences in the storage and configuration of the current replicator metadata and position can be controlled without resorting to updating the corresponding database directly.

The command is driven by a number of command-specific instructions to get or set the datasource position.

Table 8.6. dsctl Commands

| Option | Description |
|---|---|
| get | Return the available position information |

| Option | Description |
|---|---|
| help | Print the help display |
| reset | Clear the datasource position information |
| set | Set the position |

These must be used in conjunction with one of the following options to select the required datasources or service:

Table 8.7. dsctl Command-line Options

| Option | Description |
|---|---|
| `-conf` | Path to the static services properties file |
| `-ds` | Name of the datasource |
| `-service` | Name of the replication service to get information from |

If more than one service or datasource has been configured, one of these options much be used to select the service. Otherwise, by default dsctl will use the corresponding configured service.

## 8.7.1. dsctl get Command

Table 8.8. dsctl Command-line Options

| Option | Description |
|---|---|
| `-ascmd` | Generates the command required to set the datasource to the current position |

Returns the current datasource status and position, returning the information as a JSON string. The example below has been formatted for clarity:

```
shell> dsctl
[
  {
    "last_frag" : true,
    "applied_latency" : 1,
    "extract_timestamp" : "2015-01-21 21:46:57.0",
    "eventid" : "mysql-bin.000014:0000000000005645;-1",
    "source_id" : "tr-11",
    "epoch_number" : 22,
    "update_timestamp" : "2015-01-21 21:46:58.0",
    "task_id" : 0,
    "shard_id" : "tungsten_alpha",
    "seqno" : 22,
    "fragno" : 0
  }
]
```

When the `-ascmd` option is used, the information is output in form of a command:

```
shell> dsctl get -ascmd
dsctl set -seqno 17 -epoch 11 -event-id "mysql-bin.000082:0000000014031577;-1" -source-id "ubuntu"
```

If the `-reset` is used, then the generated command also includes the option. For example:

```
shell> dsctl get -ascmd -reset
dsctl set -seqno 17 -epoch 11 -event-id "mysql-bin.000082:0000000014031577;-1" -source-id "ubuntu" -reset
```

## 8.7.2. dsctl set Command

Table 8.9. dsctl Command-line Options

| Option | Description |
|---|---|
| `-epoch` | Epoch Number |
| `-event-id` | Source Event ID |
| `-reset` | Resets the datasources before performing set operation |
| `-seqno` | Sequence number |
| `-source-id` | Source ID |

Sets the current replicator position. When using this option, the `-seqno`, `-epoch`, `-event-id`, and `-source-id` options must be specified to set the corresponding values in the replicator.

For example:

```
shell> dsctl set -seqno 22 -epoch 22 -event-id "mysql-bin.000014:0000000000005645;-1" -source-id tr-11
Service "alpha" datasource "global" position was set to: seqno=22 epoch_number=22 »
    eventid=mysql-bin.000014:0000000000005645;-1 source_id=tr-11
```

When used with the `-reset`, the datasource is reset before the set operation:

```
shell> dsctl set -seqno 17 -epoch 11 -event-id "mysql-bin.000082:0000000014031577;-1" -source-id "ubuntu" -reset
Service "alpha" datasource "global" catalog information cleared
Service "alpha" datasource "global" position was set to: seqno=17 epoch_number=11 »
    eventid=mysql-bin.000082:0000000014031577;-1 source_id=ubuntu
```

Adding the `-reset` option to the dsctl get -ascmd command also adds the option to the generated command:

```
shell> dsctl get -ascmd -reset
dsctl set -seqno 17 -epoch 11 -event-id "mysql-bin.000082:0000000014031577;-1" -source-id "ubuntu" -reset
```

## 8.7.3. dsctl reset Command

Clears the current replicator status and position information:

```
shell> dsctl reset
Service "alpha" datasource "global" catalog information cleared
```

## 8.7.4. dsctl help Command

Displays the current help text:

```
shell> dsctl help
Datasource Utility
Syntax: dsctl [conf|service] [-ds name] [operation]
Configuration (required if there's more than one service):
  -conf path       - Path to a static-<svc>.properties file
     OR
  -service name    - Name of a replication service to get datasource configuration from
Options:
 [-ds name]        - Name of the datasource (default: global)
Operations:
  get              - Return all available position information
     [-ascmd]      - Return all available position information as command
  set -seqno ###   - Set position (all four parameters required)
      -epoch ###
      -event-id AAAAAAAAA.######:#######
      -source-id AAA.AAA.AAA
     [-reset]      - Optionally reset the information first
  reset            - Clear datasource position information
  help             - Print this help display
```

## 8.8. `env.sh` Script

After installation, the `env.sh` can be used to setup the local environment, such as appending to the local $PATH.

If --profile-script is set during installation, then the local profile script will also be updated to ensure the `env.sh` file is loaded at login of the OS user.

```
shell> cat .bash_profile
...
# Begin Tungsten Environment for /opt/continuent
# Include the Tungsten variables
# Anything in this section may be changed during the next operation
if [ -f "/opt/continuent/share/env.sh" ]; then
    . "/opt/continuent/share/env.sh"
fi
# End Tungsten Environment for /opt/continuent
```

If not set, then the script can manually be sourced

```
shell> source /opt/continuent/share/env.sh
```

If --executable-prefix is set, then the `env.sh` script will also configure aliases for all of the common executable binaries

For example, if --executable-prefix has been set to "mm", then aliases for executable binaries will be prefixed with this value, as shown in the small example below:

```
shell> alias
...
alias mm_thl='/opt/continuent/tungsten/tungsten-replicator/bin/thl'
alias mm_tpm='/opt/continuent/tungsten/tools/tpm'
alias mm_trepctl='/opt/continuent/tungsten/tungsten-replicator/bin/trepctl'
...
```

# 8.9. The load-reduce-check Tool

The load-reduce-check tool provides a single command to perform the final steps to convert data loaded through the Hadoop applier into a final, Hive-compatible table providing a carbon copy of the data within Hive as extracted from the source database.

See Section 4.6, "Deploying the Hadoop Applier" for more details on configuring the Hadoop Applier.

The four steps, each of which can be enabled or disabled individually are:

1. Section 8.9.1, "Generating Staging DDL"

   Accesses the source database, reads the schema definition, and generates the necessary DDL for the staging tables within Hive. Tables are by default prefixed with `stage_xxx_`, and created in a Hive schema matching the source schema.

2. Section 8.9.2, "Generating Live DDL"

   Accesses the source database, reads the schema definition, and generates the necessary DDL for the tables within Hive. Tables are created with an identical table and schema name to the source schema.

3. Section 8.9.3, "Materializing a View"

   Execute a view materialization, where the data in any existing table, and the staging table are merged into the final table data. This step is identical to the process executed when running the materialize tool.

4. Section 8.9.6, "Compare Loaded Data"

   Compares the data within the source and materialized tables and reports any differences.

The load-reduce-check tool

## 8.9.1. Generating Staging DDL

## 8.9.2. Generating Live DDL

## 8.9.3. Materializing a View

## 8.9.4. Generating Sqoop Load Commands

## 8.9.5. Generating Metadata

## 8.9.6. Compare Loaded Data

# 8.10. The materialize Command

# 8.11. The tungsten_merge_logs Script

The tungsten_merge_logs command is designed to aid troubleshooting by consolidating the various log files into one place ordered by time.

`tungsten_merge_logs` [ `--after {TIMESTAMP}` ] [ `--before {TIMESTAMP}` ] [ `--debug`, `--d` ] [ `--extension`, `--X` ] [ `--help`, `-h` ] [ `--log-limit`, `--L` ] [ `--quiet`, `--q` ] [ `--replicator`, `--R` ] [ `--stdout`, `--O` ] [ `--verbose`, `-v` ]

Where:

Table 8.10. tungsten_merge_logs Command-line Options

| Option | Description |
|---|---|
| `--after {TIMESTAMP}` | Discard all lines prior to {TIMESTAMP} |
| `--before {TIMESTAMP}` | Discard all lines after {TIMESTAMP} |
| `--debug`, `--d` | |
| `--extension`, `--X` | Specify the file extension (default: log) Do NOT include the period |
| `--help`, `-h` | Show help text |
| `--log-limit`, `--L` | Specify the quantity of log files to gather (default: unlimited). When the wrapper rotates log files, it appends a period and an integer to the end of the log file name, when .1 is the newest and .2 is older and .3 older than that, etc. This parameter will gather the base log file plus limit minus one rotated files. |
| `--quiet`, `--q` | |
| `--replicator`, `--R` | Include the replicator log files |
| `--stdout`, `--O` | Send merged logs to STDOUT instead of a file |
| `--verbose`, `-v` | Show verbose output |

With no options specified, the tungsten_merge_logs script will gather all log files in the current directory and below.

For example:

```
shell> cd
shell> tpm diag --all
shell> tar xvzf ungsten-diag-2021-11-15-16-37-33.tgz
shell> cd tungsten-diag-2021-11-15-16-37-33
shell> tungsten_merge_logs
```

Would result in something like the following:

```
New merged log file ./merged.log created!
```

All logs files are gathered by default.

Using multiple options will aggregate the logs from the specified components.

Use of the `--log-limit` option works as follows:

- a loglimit of 1 means gather the base file only, i.e. trepsvc.log
- a loglimit of 2 means gather the base file and the first backup file, i.e. trepsvc.log and trepsvc.log.1
- a loglimit of 3 means gather the base file and the first two backup files, i.e. trepsvc.log, trepsvc.log.1 and trepsvc.log.2

The `{TIMESTAMP}` must be specified as a single argument wrapped in quotes, in the format of `'yyyy/mm/dd hh:mm:ss'`, including a single space between the date and time. Hours are in 24-hour time, and all values should be left-padded with zeros. For example:

```
shell> tungsten_merge_logs --before '2021/09/27 21:58:02'
```

# 8.12. The multi_trepctl Command

The multi_trepctl command provides unified status and operation support across your Tungsten Cluster installation across multiple hosts without the need to run the trepctl command across multiple hosts and/or services individually.

`multi_trepctl`
masterof
backups [ `--by-service` ] [ `--fields appliedLastSeqNo` | `appliedLatency` | `host` | `role` | `serviceName` | `state` ]
heartbeat [ `--host`, `--hosts self` ]
list [ `--output json` | `list` | `name` | `tab` | `yaml` ] [ `--path`, `--paths` ] [ `--role`, `--roles` ]
run [ `--service`, `--services self` ] [ `--skip-headers` ] [ `--sort-by` ]

The default operation, with no further command-line commands or arguments displays the status of all the hosts and services identified as related to the current host. In a typical single-service deployment, the command outputs the status of all services by determining the relationship between hosts connected to the default service:

```
shell> multi_trepctl
| host   | serviceName | role   | state   | appliedLastSeqno | appliedLatency |
| tr-ms1 | alpha       | master | ONLINE  |               54 |          0.867 |
| tr-ms2 | alpha       | slave  | ONLINE  |               54 |          1.945 |
| tr-ms3 | alpha       | slave  | ONLINE  |               54 |         42.051 |
```

On a server with multiple services, information is output for each service and host:

```
shell> multi_trepctl
| host  | servicename | role   | state         | appliedlastseqno | appliedLatency |
| east1 | east        | master | ONLINE        |               53 |          0.000 |
| east1 | west        | slave  | OFFLINE:ERROR |               -1 |         -1.000 |
| west1 | west        | master | ONLINE        |           294328 |          0.319 |
| west1 | east        | slave  | ONLINE        |               53 |        119.834 |
| west2 | west        | master | ONLINE        |           231595 |          0.316 |
| west2 | east        | slave  | ONLINE        |               53 |        181.128 |
| west3 | east        | slave  | ONLINE        |               53 |        204.790 |
| west3 | west        | slave  | ONLINE        |           231595 |         22.895 |
```

## 8.12.1. multi_trepctl Options

The multi_trepctl tool provides a number of options that control the information and detail output when the command is executed.

### Table 8.11. multi_trepctl Command-line Options

| Option | Description |
|---|---|
| `--by-service` [257] | Sort the output by the service name |
| `--fields` [257] | Fields to be output during during summary |
| `--host` [257], `--hosts` [257] | Host or hosts on which to limit output |
| `--output` [258] | Specify the output format |
| `--paths` [258], `--path` [258] | Directory or directories to check when looking for tools |
| `--role` [258], `--roles` [258] | Role or roles on which to limit output |
| `--service` [259], `--services` [259] | Service or services on which to limit output |
| `--skip-headers` [259] | Skip the headers |
| `--sort-by` [259] | Sort by a specified field |

Where:

- `--by-service` [257]

  Order the output according to the service name and role within the service:

```
shell> multi_trepctl --by-service
| host  | servicename | role   | state         | appliedlastseqno | appliedLatency |
| east1 | east        | master | ONLINE        |               64 |         59.380 |
| west1 | east        | slave  | ONLINE        |               64 |         60.889 |
| west2 | east        | slave  | ONLINE        |               64 |         60.970 |
| west3 | east        | slave  | ONLINE        |               64 |         61.097 |
| west1 | west        | master | ONLINE        |           294328 |          0.319 |
| west2 | west        | master | ONLINE        |           231595 |          0.316 |
| east1 | west        | slave  | OFFLINE:ERROR |               -1 |         -1.000 |
| west3 | west        | slave  | ONLINE        |           231595 |         22.895 |
```

- `--fields` [257]

  Limited the output to the specified list of fields from the output of fields output by trepctl. For example, to limit the output to the host, role, and *`appliedLatency`*:

```
shell> multi_trepctl --fields=host,role,appliedlatency
| host   | role   | appliedlatency |
| tr-ms1 | master |          0.524 |
| tr-ms2 | slave  |          0.000 |
| tr-ms3 | slave  |         -1.000 |
```

- `--host` [257], `--hosts` [257]

  Limit the output to the host, or a comma-separated list of hosts specified. For example:

```
shell> multi_trepctl --hosts=tr-ms1,tr-ms3
| host   | servicename | role   | state         | appliedlastseqno | appliedlatency |
| tr-ms1 | alpha       | master | ONLINE        |             2322 |          0.524 |
```

```
| tr-ms3 | alpha       | slave  | OFFLINE:ERROR |               -1 |          -1.000 |
```

- `--output` [258]

  Specify the output format.

  Table 8.12. multi_trepctl `--output` Option

| Option | `--output` [258] | |
|---|---|---|
| Description | Specify the output format | |
| Value Type | string | |
| Default | info | |
| Valid Values | json | JSON format |
| | list | List format |
| | name | Name (simplified text) format |
| | tab | Tab-delimited format |
| | yaml | YAML format |

  For example, to output the current status in JSON format:

```
shell> multi_trepctl --output json
[
  {
    "appliedlastseqno": 2322,
    "appliedlatency": 0.524,
    "host": "tr-ms1",
    "role": "master",
    "servicename": "alpha",
    "state": "ONLINE"
  },
  {
    "appliedlastseqno": 2322,
    "appliedlatency": 0.0,
    "host": "tr-ms2",
    "role": "slave",
    "servicename": "alpha",
    "state": "ONLINE"
  },
  {
    "appliedlastseqno": -1,
    "appliedlatency": -1.0,
    "host": "tr-ms3",
    "role": "slave",
    "servicename": "alpha",
    "state": "OFFLINE:ERROR"
  }
]
```

- `--path` [258], `--paths` [258]

  Limit the search for trepctl to the specified path or comma-separated list of paths. On a deployment with multiple services, the output will be limited by the services installed within the specified directories:

```
shell> multi_trepctl --path /opt/replicator
| host | servicename | role  | state  | appliedlastseqno | appliedlatency |
| db1  | west        | slave | ONLINE |                3 |          0.450 |
| db2  | west        | slave | ONLINE |                3 |          0.481 |
| db3  | west        | slave | ONLINE |                3 |          0.484 |
| db4  | east        | slave | ONLINE |                4 |          0.460 |
| db5  | east        | slave | ONLINE |                4 |          0.451 |
| db6  | east        | slave | ONLINE |                4 |          0.496 |
```

  This is also useful when control of cross-site replicators is desired in MSMM topologies prior to v6.0.0.

  For example, take all cross-site replicators offline:

```
shell> multi_trepctl --path /opt/replicator offline
```

  To bring all cross-site replicators online:

```
shell> multi_trepctl --path /opt/replicator online
```

- `--role` [258], `--roles` [258]

Limit the output to show only the specified role or comma-separated list of roles:

```
shell> multi_trepctl --roles=slave
| host   | servicename | role  | state         | appliedlastseqno | appliedlatency |
| tr-ms2 | alpha       | slave | ONLINE        |             2322 |          0.000 |
| tr-ms3 | alpha       | slave | OFFLINE:ERROR |               -1 |         -1.000 |
```

- `--service` [259], `--services` [259]

  Limit the output to the specified service or comma-separated list of services:

```
shell>  multi_trepctl --service=east
| host  | servicename | role   | state         | appliedlastseqno | appliedlatency |
| east1 | east        | master | ONLINE        |               53 |          0.000 |
| west1 | east        | slave  | ONLINE        |               53 |        119.834 |
| west2 | east        | slave  | ONLINE        |               53 |        181.128 |
| west3 | east        | slave  | ONLINE        |               53 |        204.790 |
```

- `--skip-headers` [259]

  Prevents the generation of the headers when generating the list output format:

```
shell> multi_trepctl --skip-headers
| tr-ms1 | alpha | master | ONLINE        | 2322 | 0.524 |
| tr-ms2 | alpha | slave  | ONLINE        | 2322 | 0.000 |
| tr-ms3 | alpha | slave  | OFFLINE:ERROR |   -1 | -1.000 |
```

- `--sort-by` [259]

  Sort by the specified fieldname. For example, to sort the output by the latency:

```
shell> multi_trepctl --sort-by appliedlatency
| host   | servicename | role   | state         | appliedlastseqno | appliedlatency |
| tr-ms3 | alpha       | slave  | OFFLINE:ERROR |               -1 |         -1.000 |
| tr-ms2 | alpha       | slave  | ONLINE        |             2322 |          0.000 |
| tr-ms1 | alpha       | master | ONLINE        |             2322 |          0.524 |
```

## 8.12.2. multi_trepctl Commands

The default operational mode is for multi_trepctl list to output the status. A specific mode can be also be specified on the command-line.

### Table 8.13. multi_trepctl Commands

| Option | Description |
|---|---|
| masterof | List all the Primaries of configured hosts and services |
| backups | List all the backups available across all configured hosts and services |
| heartbeat | Inserts a heartbeat on all Primaries within the service |
| list | List the information about each service |
| run | Run the specified trepctl command on all hosts/services |

In addition to the two primary commands, multi_trepctl can execute commands that would normally be applied to trepctl, running them on each selected host, service or directory according to the options. The output format and expectation is controlled through the list and run commands.

For example:

```
shell> multi_trepctl status
```

Outputs the long form of the status information (as per trepctl status) for each identified host.

### 8.12.2.1. multi_trepctl backups Command

Lists the available backups across all replicators.

```
shell> multi_trepctl backups
| host  | servicename | backup_date         | prefix           | agent     |
| host1 | alpha       | 2014-08-15 09:40:37 | store-0000000002 | mysqldump |
| host1 | alpha       | 2014-08-15 09:36:57 | store-0000000001 | mysqldump |
| host2 | alpha       | 2014-08-12 07:02:29 | store-0000000001 | mysqldump |
```

### 8.12.2.2. multi_trepctl heartbeat Command

Runs the trepctl heartbeat command on all hosts that are identified as masters.

```
shell> multi_trepctl heartbeat
host: host1
servicename: alpha
role: master
state: ONLINE
appliedlastseqno: 8
appliedlatency: 2.619
output:
```

### 8.12.2.3. multi_trepctl masterof Command

Lists which hosts are Primaries of others within the configured services.

```
shell> multi_trepctl masterof
| servicename | host  | uri               |
| alpha       | host1 | thl://host1:2112/ |
```

### 8.12.2.4. multi_trepctl list Command

The multi_trepctl list mode is the default mode for multi_trepctl and outputs the current status across all hosts and services as a table:

```
shell> multi_trepctl
| host  | servicename | role   | state                      | appliedlastseqno | appliedlatency |
| host1 | firstrep    | master | OFFLINE:ERROR              |               -1 |         -1.000 |
| host2 | firstrep    | slave  | GOING-ONLINE:SYNCHRONIZING |             5271 |       4656.264 |
| host3 | firstrep    | slave  | OFFLINE:ERROR              |               -1 |         -1.000 |
| host4 | firstrep    | slave  | OFFLINE:ERROR              |               -1 |         -1.000 |
```

Or selected hosts and services if options are specified. For example, to get the status only for `host1` and `host2`:

```
shell> multi_trepctl --hosts=host1,host2
| host  | servicename | role   | state  | appliedlastseqno | appliedlatency |
| host1 | firstrep    | master | ONLINE |             5277 |          0.476 |
| host2 | firstrep    | slave  | ONLINE |             5277 |          0.000 |
```

The multi_trepctl command implies that the status or information is being output from each of the commands executed on the remote hosts and services.

### 8.12.2.5. multi_trepctl run Command

The multi_trepctl run command can be used where the output of the corresponding trepctl command cannot be formatted into a convenient list. For example, to execute a backup on every host within a deployment:

```
shell> multi_trepctl run backup
```

The same filters and host or service selection can also be made:

```
shell> multi_trepctl run backup --hosts=host1,host2,host3
host: host1
servicename: firstrep
output: |
  Backup completed successfully; URI=storage://file-system/store-0000000005.properties
---
host: host2
servicename: firstrep
output: |
  Backup completed successfully; URI=storage://file-system/store-0000000001.properties
---
host: host3
servicename: firstrep
output: |
  Backup completed successfully; URI=storage://file-system/store-0000000001.properties
...
```

Return from the command will only take place when remote commands on each host have completed and returned.

## 8.13. The tungsten_newrelic_event Command

The tungsten_newrelic_event script utilises existing tunsten monitor scripts and inserts the results into New Relic.

By default all of the following Nagios check scripts under `$CONTINUENT_ROOT/tungsten/cluster-home/bin` are executed and the results of each are inserted into NewRelic as the associated EventType.

| Executable | EventType |
|---|---|
| check_tungsten_latency | CheckTungstenLatency |
| check_tungsten_online | CheckTungstenOnline |
| check_tungsten_policy | CheckTungstenPolicy |
| check_tungsten_progress | CheckTungstenProgress |
| check_tungsten_services -r -c | CheckTungstenServices |
| check_tungsten_services -r | CheckTungstenNode |
| check_tungsten_services -c | CheckTungstenConnector |

If you specify a check to execute by using one or more cli args, then only those checks specified will be run.

curl must be installed and available on the PATH.

You must provide your New Relic Account ID and your New Relic Insights API Insert Key for the script to function.

You may obtain the API Insert Key at https://insights.newrelic.com/accounts/{New Relic Account ID}/manage/api_keys

`tungsten_monitor` [ `--account` ] [ `--critical {seconds}`, `-C {seconds}` ] [ `--curl` ] [ `--debug`, `-d` ] [ `--help`, `-h` ] [ `--hostname` ] [ `--key` ] [ `--latency`, `-l` ] [ `--noexec` ] [ `--online`, `-o` ] [ `--progress`, `-r` ] [ `--service {SERVICE}` ] [ `--test` ] [ `--timeout {seconds}`, `-t {seconds}` ]
tmpfile [ `--node`, `-n` ] [ `--services`, `-s` ] [ `--verbose`, `--v` ] [ `--warn {seconds}`, `-W {seconds}` ]

Where:

### Table 8.14. tungsten_monitor Command-line Options

| Option | Description |
|---|---|
| `--account` | Use to spceify your New Relic Account ID |
| `-C {seconds}`, `--critical {seconds}` | [default: 15] Specify the Critical alert latency level in seconds; optionally used for check_tungsten_latency |
| `--curl` | Use to specify full path the curl binary |
| `--debug`, `-d` | Enabling debug also implies enabling of verbose. Debug is VERY chatty, avoid unless essential |
| `--help`, `-h` | |
| `--hostname` | Use to specify full path the hostname binary |
| `--key` | Use to specify your New Relic Insights API Insert key |
| `--latency`, `-l` | Run check_tungsten_latency, optionally specify --warn and --crit |
| `--noexec` | Do not execute the Tungsten Nagios Check script |
| `--online`, `-o` | Run check_tungsten_online, optionally specify a service with --service |
| `--progress`, `-r` | Run check_tungsten_progress, optionally specify a timeout with --timeout, optionally specify a service with --service |
| `--service {SERVICE}` | Specify a service name; optionally used for check_tungsten_online and check_tungsten_progress |
| `--test` | Do not execute the New Relic API Insert call |
| `-t {seconds}`, `--timeout {seconds}` | [default: 1] Specify the time to wait for progress to occur; optionally used for check_tungsten_progress |
| `tmpfile` | Use to specify full path the temp JSON file. Default: ./new_relic_event.json |
| `-n`, `--node` | Run check_tungsten_services -r to check for the Replicator |
| `-s`, `--services` | Run check_tungsten_services -r to check for the Replicator |
| `--verbose`, `--v` | |
| `-W {seconds}`, `--warn {seconds}` | [default: 5] Specify the Warning alert latency level in seconds; optionally used for check_tungsten_latency |

Usage

```
shell> tungsten_newrelic_event --account {New Relic Account ID} --key {New Relic Insights API Insert Key} [args]
```

# 8.14. The query Command

Table 8.15. query Common Options

| Option | Description |
|---|---|
| `-conf PATH` | Configuration file that contains values for connection properties (url, user and password) |
| `-file PATH` | File containing the SQL commands to run. If missing, read SQL commands from STDIN |
| `-password` | Prompt for password |
| `-url JDBCURL` | JDBC url of the database to connect to |
| `-user USER` | User used to connect to the database |

The query command line tool can be used to issue SQL statements against a database.

The queries can either be entered via STDIN, or read in from a text file

The following example shows a SELECT statement issued via STDIN

```
shell> query -url "jdbc:mysql:thin://db2:13306/" -user tungsten -password
Enter password: ********
select * from tungsten_nyc.trep_commit_seqno;

[
 {
  "statement":"select * from tungsten_nyc.trep_commit_seqno;","rc":0,"results":
 [
  [
   {
    "task_id":0,
    "seqno":1,
    "fragno":0,
    "last_frag":"1",
    "source_id":"db1",
     "epoch_number":1,
    "eventid":"mysql-bin.000002:0000000000000879;-1",
    "applied_latency":0,
    "update_timestamp":"2019-06-28 10:44:20.0",
    "shard_id":"tungsten_nyc",
    "extract_timestamp":"2019-06-28 10:44:19.0"
   }
  ]
 ],
  "error":null
 }
]
```

# 8.15. The replicator Command

The replicator is the wrapper script that handles the execution of the replicator service.

Table 8.16. replicator Commands

| Option | Description |
|---|---|
| `condrestart` [263] | Restart only if already running |
| `console` [263] | Launch in the current console (instead of a daemon) |
| `dump` [263] | Request a Java thread dump (if replicator is running) |
| `install` [263] | Install the service to automatically start when the system boots |
| `remove` [263] | Remove the service from starting during boot |
| `restart` [263] | Stop replicator if already running and then start |
| `start` [264] | Start in the background as a daemon process |
| `status` [264] | Query the current status |

| Option | Description |
|---|---|
| stop [264] | Stop if running (whether as a daemon or in another console) |

These commands and options are described below:

– condrestart [263]

### Table 8.17. replicator Commands Options for condrestart [263]

| Option | Description |
|---|---|
| offline | Start in OFFLINE state |

Restart the replicator, only if it is already running. This can be useful to use when changing configuration or performing database management within automated scripts, as the replicator will be only be restart if it was previously running.

For example, if the replicator is running, replicator condrestart operates as replicator restart:

```
shell> replicator condrestart
Stopping Tungsten Replicator Service...
Waiting for Tungsten Replicator Service to exit...
Stopped Tungsten Replicator Service.
Starting Tungsten Replicator Service...
Waiting for Tungsten Replicator Service......
running: PID:26646
```

However, if not already running, the operation does nothing:

```
shell> replicator condrestart
Stopping Tungsten Replicator Service...
Tungsten Replicator Service was not running.
```

– console [263]

### Table 8.18. replicator Commands Options for console [263]

| Option | Description |
|---|---|
| offline | Start in OFFLINE state |

Launch in the current console (instead of a daemon)

– dump [263]

Request a Java thread dump (if replicator is running)

– install [263]

Installs the startup scripts for running the replicator at boot. For an alternative method of deploying these start-up scripts, see deployall.

– remove [263]

Removes the startup scripts for running the replicator at boot. For an alternative method of removing these start-up scripts, see undeployall.

– restart [263]

### Table 8.19. replicator Commands Options for restart [263]

| Option | Description |
|---|---|
| offline | Stop and restart in OFFLINE state |

#### Warning

Restarting a running replicator temporarily stops and restarts replication.

Stops the replicator, if it is already running, and then restarts it:

```
shell> replicator restart
Stopping Tungsten Replicator Service...
Stopped Tungsten Replicator Service.
Starting Tungsten Replicator Service...
```

```
Waiting for Tungsten Replicator Service......
running: PID:26248
```

– start [264]

Table 8.20. replicator Commands Options for start [264]

| Option | Description |
|---|---|
| offline | Start in OFFLINE state |

To start the replicator service if it is not already running:

```
shell> replicator start
 Starting Tungsten Replicator Service...
```

– status [264]

Checks the execution status of the replicator:

```
shell> replicator status
Tungsten Replicator Service is running: PID:27015, Wrapper:STARTED, Java:STARTED
```

If the replicator is not running:

```
shell> replicator status
Tungsten Replicator Service is not running.
```

This only provides the execution state of the replicator, not the actual state of replication. To get detailed information on the status of replication use trepctl status.

– stop [264]

Stops the replicator if it is already running:

```
shell> replicator stop
Stopping Tungsten Replicator Service...
Waiting for Tungsten Replicator Service to exit...
Stopped Tungsten Replicator Service.
```

# 8.16. The startall Command

The startall will start all configured services within the configured directory:

```
Starting Tungsten Replicator Service...
Waiting for Tungsten Replicator Service......
running: PID:29842
```

If a service is already running, then a notification of the current state will be provided:

```
Starting Tungsten Replicator Service...
Tungsten Replicator Service is already running.
```

Note that if any service is not running, and a suitable PID is found, the file will be deleted and the services started, for example:

```
Removed stale pid file:
 /opt/continuent/releases/tungsten-replicator-7.1.2-81_pid25898/tungsten-connector/bin/../var/tconnector.pid
```

# 8.17. The stopall Command

The stopall command stops all running services if they are already running:

```
shell> stopall
Stopping Tungsten Replicator Service...
Waiting for Tungsten Replicator Service to exit...
Stopped Tungsten Replicator Service.
```

# 8.18. The tapi Command

The tapi command was added in version 7.0.0

The tapi command is designed to act as an interface to the Tungsten REST APIv2.

The original goal of this script was to provide a way to test and exercise the API which then grew to include additional functionality and convenience options.

Usage for tapi:

```
shell> tapi [options] [API_call_path]
```

See the tables below for a list of valid arguments:

### Table 8.21. tapi Generic Options

| Option | Description |
|---|---|
| `--debug`, `-d` | |
| `--help`, `-h` | |
| `--quiet`, `-q` | |
| `--service`, `-s` | Specify a service name. Used with --run, --getpolicy, --setpolicy and the --policy NRPE plugin |
| `--verbose`, `-v` | |

### Table 8.22. tapi CURL-related Options

| Option | Description |
|---|---|
| `--curl` | Specify where to find curl executable (default: curl) |
| `--get` | Set the curl request to GET |
| `--host` | Specify the host name to connect to (default: 127.0.0.1) |
| `--nossl`, `--http` | Specify NON-SSL/http-only connection (default: https, SSL enabled) |
| `--port` | Specify the port number to connect to (default: 8090) |
| `--post` | Set the curl request to POST |
| `--put` | Set the curl request to PUT |
| `-p`, `--password` | Specify auth password (default: not defined) |
| `--request {GET\|POST\|PUT}` | Specify the curl request method (default: none) |
| `--user`, `-u` | Specify auth user (default: not defined) |

### Table 8.23. tapi Nagios/NRPE/Zabbix-related Options

| Option | Description |
|---|---|
| `--allperf`, `--per-replica-perfdata` | When --allperf is specified with --perf, each service is listed separately; used with --latency |
| `-c`, `--critical` | Specify the critical threshold value in seconds; used with --latency |
| `--latency` | Check the replicator for latency as an NRPE plugin |
| `--online` | Check the manager to see if the datasource is online as an NRPE plugin |
| `--perf`, `--perfdata` | Display additional performance data; used with --latency |
| `--policy` | Check the manager to see if the cluster is in automatic policy mode as an NRPE plugin |
| `--progress` | Check the replicator to see that it is making progress over time as an NRPE plugin |
| `--services` | Check the host for running services (Manager, Connector and Replicator) as an NRPE plugin |
| `--wait`, `-W` | Specify the time to wait in seconds during the --progress test (default: 1 second) |
| `--warn`, `-w` | Specify the warning threshold value in seconds; used with --latency |
| `--zabbix`, `-z` | Use Zabbix-style exit messages which are the same as the exit code, unlike NRPE which outputs a string starting with OK, WARNING or CRITICAL |

Table 8.24. tapi Admin-related Options

| Option | Description |
|---|---|
| `--create` | Connect to the specified host and port, then call the createAdminUser path with a special body to instantiate the user. Must supply --create-user and --create-password. |
| `--create-pass` | Sets password for the APIv2 admin user created using the --create flag. --create and --create-user must also be supplied. |
| `--create-user` | Sets the user name for the APIv2 admin user created using the --create flag. --create and --create-pass must also be supplied. |
| `--getds` | Display the main Manager dataservice name |
| `--getmain` | Display the main Replicator service name |
| `--getpolicy` | Get the current local policy. If --service {svcname} is specified, then the policy will be displayed for that service. |
| `--getports` | Display the API default port for each component |
| `--getsubs` | Display the Replicator sub-service(s) |
| `--includeServiceName` | Output service names. |
| `--no-connectors` | Append '?no-connectors=1' to all API calls to disable the Manager internal status calls to the Connectors |
| `--setauto` | Set the current policy to AUTOMATIC |
| `--setmaint` | Set the current policy to MAINTENANCE |
| `--setpolicy {auto\|maint}` | Set the current local policy. If --service {svcname} is specified, then the policy will be set for that service. One-letter abbreviations are allowed for the policy mode names. |
| `--topology` | Display the main Manager topology |

Table 8.25. tapi Filter-related Options

| Option | Description |
|---|---|
| `--connector`, `-C` | Specify the connector path |
| `--filter`, `-F` | Specify {string} to Limit the call path (typically used with --listapi) |
| `--manager`, `-M` | Specify the manager path |
| `--replicator`, `-R` | Specify the replicator path |

Table 8.26. tapi API-related Options

| Option | Description |
|---|---|
| `--allservices` | Operate on all available services. Used with --run |
| `--follow`, `--descend`, `-f` | If the resulting payload type is URIsPayload then call every key in the payload as a sub-item of the original path |
| `--listapi`, `-L` | List all available REST API calls by unique key |
| `--run`, `--alias`, `-r` | Specify an API call name by unique key. Use --listapi to see all keys/aliases |

Table 8.27. tapi Status-related Options

| Option | Description |
|---|---|
| `--affinity` | Display only the affinity-specific values from the '/connector/configuration/module/connector' path |
| `--allstats` | Display all statistics cluster-wide using the Manager API |
| `--connectorstatus` | Use the '/connector/configuration/module/connector' path |
| `--cs`, `--clusterstatus` | Use the 'manager/cluster/status' path |
| `--mgrstats`, `--managerstats` | Display Manager statistics cluster-wide using the Manager API |
| `--routers` | Display Connector statistics cluster-wide using the Manager API |

| Option | Description |
|---|---|
| `--status` | Use the 'manager/status' path |
| `--trstats` | Display Replicator statistics cluster-wide using the Manager API |

Table 8.28. tapi Backup and Restore-related Options

| Option | Description |
|---|---|
| `--backuptimeout` | Specify the backup timeout in seconds |
| `--mysqldump` | Use the mysqldump backup agent |
| `--restoretimeout` | Specify the restore timeout in seconds |
| `--restoreuri` | Specify the restore URI |
| `--storageagent` | Specify the storage agent |
| `--xtrabackup` | Use the xtrabackup backup agent |
| `--xtrabackupFull` | Use the xtrabackupFull backup agent |
| `--xtrabackupIncremental` | Use the xtrabackupIncremental backup agent |

There are many cli options which invoke the various functions of the tapi script.

With no options specified, the tapi script will construct a curl command to query the Tungsten Manager API (default https://127.0.0.1:8090) using the {API_call_path} provided on the command line.

Basic Example

Running tapi -M status would result in something like the following:

```
shell> tapi manager/status
Executing: curl -s --user tungsten:secret --insecure --request GET 'https://127.0.0.1:8090/api/v2/manager/status'
{"payloadType":"StatusPayload","payloadVersion":"1","payload":{"dataServiceName":"east","dataSourceName":"db1-demo.continuent.com",
"startTime":"2021-03-25T16:47:50.523 UTC","uptimeSeconds":1909,"state":"ONLINE","isCoordinator":true,"isWitness":false,
"managerPID":4052,"parentPID":4031,"policyMode":"MAINTENANCE","coordinator":"db1-demo.continuent.com"}}
```

If both `--user` and `--password` are defined, curl will use them in the call.

If either or both `--user` and `--password` are missing, tapi will attempt to derive the values using the tpm command.

Simple Admin Command Examples

```
tapi --getds
tapi --topology
tapi --setmaint
tapi --setauto

tapi --setpolicy maintenance
tapi --setpolicy m
tapi --setpolicy automatic
tapi --setpolicy a
```

Simple Replicator Command Examples

```
tapi --getmain
tapi --getsubs
```

Running API Command Examples

Show all available API calls in four columns: component, unique key, request type, request path:

```
tapi -R --listapi
```

Show all available API calls filtered by Manager (-M), Replicator (-R) or Connector (-c):

```
tapi --listapi -M
tapi --listapi -R
tapi --listapi -C
```

Execute an API call by unique key per component:

```
tapi -R --listapi
tapi -R --run ping
tapi -R -v --run offline (like `trepctl offline`)
```

```
tapi -R -v --run online (like `trepctl online`)
```

Backup and Restore Examples

```
tapi -v -R --run backup --mysqldump
tapi -v -R --run backup --xtrabackupFull
tapi -v -R -j --run restore
tapi -v -R -j --run task d28465a2-6023-47c4-9a4c-20f93514db75
```

# 8.19. The thl Command

The thl command provides an interface to the THL data, including the ability to view the list of available files, details of the enclosed event information, and the ability to purge THL files to reclaim space on disk beyond the configured log retention policy.

The command supports two command-line options that are applicable to all operations, as shown in Table 8.29, "thl Options".

### Table 8.29. thl Options

| Option | Description |
|---|---|
| `-conf path` | Path to the configuration file containing the required replicator service configuration |
| `-service servicename` | Name of the service to be used when looking for THL information |

For example, to execute a command on a specific service:

```
shell> thl index -service firstrep
```

Individual operations are selected by use of a specific command to the thl command. Supported commands are:

- dsctl — obtain syntax that can be used with the dsctl command to assist in positioning of the replicator.
- index — obtain a list of available THL files.
- info — obtain summary information about the available THL data.
- list — list one or more THL events.
- purge — purge THL data.
- help — get the command help text.

Further information on each of these operations is provided in the following sections.

## 8.19.1. thl Position Commands

The thl command supports a number of position and selection command-line options that can be used to select an individual THL event, or a range of events, to be displayed.

- `-seqno #` [268]

  Valid for: thl list

  Output the THL sequence for the specific sequence number. When reviewing or searching for a specific sequence number, for example when the application of a sequence on a Replica has failed, the replication data for that sequence number can be individually viewed. For example:

  From version 5.3.3, the output also includes the filename of the THL file on disk where the THL event is located:

```
shell> thl list -seqno 15
SEQ# = 15 / FRAG# = 0 (last frag)
- FILE = thl.data.0000000001
- TIME = 2013-05-02 11:37:00.0
- EPOCH# = 7
- EVENTID = mysql-bin.000004:0000000000003345;0
- SOURCEID = host1
- METADATA = [mysql_server_id=1687011;unsafe_for_block_commit;dbms_type=mysql;»
    service=firstrep;shard=cheffy]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [##charset = UTF-8, autocommit = 1, sql_auto_is_null = 0, foreign_key_checks = 0, »
    unique_checks = 0, sql_mode = 'NO_AUTO_VALUE_ON_ZERO', character_set_client = 33, »
    collation_connection = 33, collation_server = 8]
- SCHEMA = cheffy
- SQL(0) = CREATE TABLE `access_log` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
```

```
  `userid` int(10) unsigned DEFAULT NULL,
  `datetime` int(10) unsigned NOT NULL DEFAULT '0',
...
```

If the sequence number selected contains multiple fragments, each fragment will be output. Depending on the content of the sequence number information, the information can be output containing only the header/metadata information or only the table data (row or SQL) that was contained within the fragment. See `-headers` and `-sql` for more information.

> **Note**
>
> Unsigned integers are displayed and stored in the THL as their negative equivalents, and translated to the correct unsigned type when the data is applied to the target database.

- `-low #` [269] and/or `-high #` [269]

  `-from #` [269] and/or `-to #` [269]

  Valid for: thl list, thl purge

  Specify the start (`-from` [269]) or end (`-to` [269]) of the range of sequence numbers to be output. If only `-from` [269] is specified, then all sequence numbers from that number to the end of the THL are output. If `-to` [269] is specified, all sequence numbers from the start of the available log file to the specified sequence number are output. If both numbers are specified, output all the sequence numbers within the specified range.

  For example:

  ```
  shell> thl list -from 320
  ```

  Or:

  ```
  shell> thl list -low 320
  ```

  Will output all the sequence number fragments from number 320.

  ```
  shell> thl list -to 540
  ```

  Or:

  ```
  shell> thl list -high 540
  ```

  Will output all the sequence number fragments up to and including 540.

  ```
  shell> thl list -from 320 -to 540
  ```

  Or:

  ```
  shell> thl list -low 320 -high 540
  ```

  Will output all the sequence number fragments from number 320 up to, and including, sequence number 540.

- `-first` [269]

  Valid for: thl list, thl purge

  The `-first` [269] selects only the first stored THL event. For example:

  ```
  shell> thl list -first
  SEQ# = 0 / FRAG# = 0 (last frag)
  - TIME = 2017-06-28 13:12:38.0
  - EPOCH# = 0
  ...
  ```

- `-first #` [269]

  Valid for: thl list, thl purge

  The `-first #` [269] selects the specified number of events, starting from the first event. For example:

  ```
  shell> thl list -first 5
  ```

  Would display the first five events from the stored THL.

- `-last` [269]

  Valid for: thl list, thl purge

The `-last` [269] selects only the last stored THL event. For example:

```
shell> thl list -last
SEQ# = 1601 / FRAG# = 0 (last frag)
- TIME = 2017-06-29 06:02:23.0
- EPOCH# = 1601
...
```

The use of this option can be particularly useful in the event of synchronisation or THL corruption due to a lack of diskspace. Using the thl purge command, the last THL event can be easily removed without having to work out the ranges and index information:

```
shell> thl purge -last
```

- `-last #` [270]

  Valid for: thl list, thl purge

  The `-last #` [270] selects the specified number of events, starting from the last-# event. For example:

  ```
  shell> thl list -last 5
  ```

  When the THL index contains events from 1558-1601, would display events 1597 through to 1601.

## 8.19.2. thl dsctl Command

The dsctl command to the thl command outputs a dsctl command that can be executed against a replicator to assist in repositioning.

The command is displayed only, it is not executed

`thl` dsctl
[-seqno # ]
[-event # ]

- `-event eventid` [270]

  Output a dsctl command for the given eventid, for example:

  ```
  shell> thl dsctl -event mysql-bin.000017:0000000074628349
  dsctl -service alpha set -reset -seqno 916 -epoch 538 -event-id "mysql-bin.000017:0000000074628349;62" -source-id "centos1"
  ```

- `-seqno #` [268]

  Output a dsctl command for the given sequence number, for example:

  ```
  shell> thl dsctl -seqno 916
  dsctl -service alpha set -reset -seqno 916 -epoch 538 -event-id "mysql-bin.000017:0000000074628349;62" -source-id "centos1"
  ```

## 8.19.3. thl list Command

The list command to the thl command outputs a list of the sequence number information from the THL. By default, the entire THL as stored on disk is output. Command-line options enable you to select individual sequence numbers, sequence number ranges, or all the sequence information from a single file.

`thl` list
[-seqno # ]
[-low # ] | [-from # ] | [-high # ] | [-to # ]
[-last] [-last #] [-first] [-first #]
[-event # ]
[-file filename ] [-no-checksum ] [-sql] [-sizes] [-sizesdetail] [-sizessummary] [-charset] [-headers] [-json] [-specs-] [-charset]

- `-event eventid` [270]

  Output THL found that matches the provided eventid. If no exact match found, a message will display details of an approximate match if found. See example below:

  An exact match is found:

  ```
  shell> thl list -event mysql-bin.000017:0000000074628349
  - METADATA = [mysql_server_id=1000;mysql_thread_id=62;unsafe_for_block_commit;dbms_type=mysql;tz_aware=true;service=alpha;shard=employees]
  - TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
  - OPTIONS = [##charset = ISO8859_1, autocommit = 1, sql_auto_is_null = 0, foreign_key_checks = 1, unique_checks = 1, sql_mode = 'NO_ENGINE_SUBSTITUTION,
  - SCHEMA = employees
  ```

```
- SQL(0) = DROP TABLE `salaries` /* generated by server */
```

No match found:

```
Event not found : Approximative match found between seqno 915 (mysql-bin.000017:0000000074628153;62) and seqno 916 (mysql-bin.000017:0000000074628349;62
```

- `-file filename` [271]

  Outputs all of the sequence number fragment information from the specified THL file. If the filename has been determined from the thl index command, or by examining the output of other fragments, the file-based output can be used to identify statements or row data within the THL.

- `-charset charset` [271]

  Specify the character set to be used to decode the character-based row data embedded within the THL event. Without this option, data is output as a hex value.

- `-hex` [271]

  For SQL that may be in different character sets, the information can be optionally output in hex format to determine the contents and context of the statement, even though the statement itself may be unreadable on the command-line.

- `-no-checksum` [271]

  Ignores checksums within the THL. In the event of a checksum failure, use of this option will enable checksums to be ignored when the THL is being read.

- `-sql`

  Prints only the SQL for the selected sequence range. Use of this option can be useful if you want to extract the SQL and execute it directly by storing or piping the output.

- `-headers`

  Generates only the header information for the selected sequence numbers from the THL. For THL that contains a lot of SQL, obtaining the headers can be used to get basic content and context information without having to manually filter out the SQL in each fragment.

  The information is output as a tab-delimited list:

```
2047 1412 0 false 2020-05-03 20:58:14.0 mysql-bin.000005:0000000579721045;0 host3
2047 1412 1 true 2020-05-03 20:58:14.0 mysql-bin.000005:0000000579721116;0 host3
2048 1412 0 false 2020-05-03 20:58:14.0 mysql-bin.000005:0000000580759206;0 host3
2048 1412 1 true 2020-05-03 20:58:14.0 mysql-bin.000005:0000000580759277;0 host3
2049 1412 0 false 2020-05-03 20:58:16.0 mysql-bin.000005:0000000581791468;0 host3
2049 1412 1 true 2020-05-03 20:58:16.0 mysql-bin.000005:0000000581791539;0 host3
2050 1412 0 false 2020-05-03 20:58:18.0 mysql-bin.000005:0000000582812644;0 host3
```

  The format of the fields output is:

```
Sequence No | Epoch | Fragment | Last | Fragment | Date/Time | EventID | SourceID | Comments
```

  For more information on the fields displayed, see Section E.1.1, “THL Format”.

- `-json`

  Only valid with the `-headers` option, the header information is output for the selected sequence numbers from the THL in JSON format. The field contents are identical, with each fragment of each THL sequence being contained in a JSON object, with the output consisting of an array of the these sequence objects. For example:

```
[
   {
      "lastFrag" : false,
      "epoch" : 7,
      "seqno" : 320,
      "time" : "2020-05-02 11:41:19.0",
      "frag" : 0,
      "comments" : "",
      "sourceId" : "host1",
      "eventId" : "mysql-bin.000004:0000000244490614;0"
   },
   {
      "lastFrag" : true,
      "epoch" : 7,
      "seqno" : 320,
      "time" : "2020-05-02 11:41:19.0",
      "frag" : 1,
```

```
      "comments" : "",
      "sourceId" : "host1",
      "eventId" : "mysql-bin.000004:0000000244490685;0"
   }
]
```

For more information on the fields displayed, see THL SEQNO [549].

- `-sizes`

  Shows the size information for a given THL event, describing either the size of the SQL, or the number of rows within the given event. For example:

```
shell> thl list -sizes
SEQ# Frag# Tstamp
...
12 0 2020-06-28 13:21:11.0  Event total: 1 chunks 73 bytes in SQL statements 0 rows
13 0 2020-06-28 13:21:10.0  Event total: 1645 chunks 0 bytes in SQL statements 1645 rows
14 0 2020-06-28 13:21:11.0  Event total: 1 chunks 36 bytes in SQL statements 0 rows
15 0 2020-06-28 13:21:11.0  Event total: 1 chunks 61 bytes in SQL statements 0 rows
16 0 2020-06-28 13:21:11.0  Event total: 1 chunks 73 bytes in SQL statements 0 rows
17 0 2020-06-28 13:21:12.0  Event total: 1 chunks 36 bytes in SQL statements 0 rows
18 0 2020-06-28 13:21:12.0  Event total: 1 chunks 61 bytes in SQL statements 0 rows
19 0 2020-06-28 13:21:10.0  Event total: 1784 chunks 0 bytes in SQL statements 1784 rows
20 0 2020-06-28 13:21:12.0  Event total: 1 chunks 73 bytes in SQL statements 0 rows
21 0 2020-06-28 13:21:11.0  Event total: 1576 chunks 0 bytes in SQL statements 1576 rows
22 0 2020-06-28 13:21:12.0  Event total: 1 chunks 36 bytes in SQL statements 0 rows
23 0 2020-06-28 13:21:12.0  Event total: 1 chunks 61 bytes in SQL statements 0 rows
...
```

  Summary information is also output identicating an overall count of the changes. For example:

```
Total ROW chunks: 69487 with 18257671 updated rows (100%)
Total STATEMENT chunks: 0 with 0 bytes (0%)
628 events processed
```

  This information can be useful when viewing or monitoring the replication progress as it can help to indicate and identify the size of a specific transaction, particularly if the transaction is large. This can be particularly useful in combination with the `-first` [269] and/or `-last` [269].

  For more detailed information on individual fragments within a sequence (and for large transactions there will be multiple fragments), use the thl list -sizesdetail command.

- `-sizesdetail`

  Shows detailed size information for a given THL event, describing either the size of the SQL, or the number of rows within the given event per fragment within each event, and with a summary for each event total. For very large THL event sizes this provide more detailed information about the size and makeup of the event. For example:

```
shell> thl list -sizes -last
SEQ# Frag# Tstamp   Chunks  SQL Data    Row Data
1604 0 2020-06-29 11:04:53.0 123 chunks SQL 0 bytes (0 avg bytes per chunk) Rows 45633 (371 avg rows per chunk)
1604 1 2020-06-29 11:04:53.0 123 chunks SQL 0 bytes (0 avg bytes per chunk) Rows 45633 (371 avg rows per chunk)
1604 2 2020-06-29 11:04:53.0 123 chunks SQL 0 bytes (0 avg bytes per chunk) Rows 45633 (371 avg rows per chunk)
1604 3 2020-06-29 11:04:53.0 123 chunks SQL 0 bytes (0 avg bytes per chunk) Rows 45633 (371 avg rows per chunk)
1604 4 2020-06-29 11:04:53.0 123 chunks SQL 0 bytes (0 avg bytes per chunk) Rows 45633 (371 avg rows per chunk)
1604 5 2020-06-29 11:04:53.0 123 chunks SQL 0 bytes (0 avg bytes per chunk) Rows 45633 (371 avg rows per chunk)
1604 6 2020-06-29 11:04:53.0 123 chunks SQL 0 bytes (0 avg bytes per chunk) Rows 45633 (371 avg rows per chunk)
1604 7 2020-06-29 11:04:53.0 123 chunks SQL 0 bytes (0 avg bytes per chunk) Rows 45633 (371 avg rows per chunk)
1604 8 2020-06-29 11:04:53.0 123 chunks SQL 0 bytes (0 avg bytes per chunk) Rows 45633 (371 avg rows per chunk)
1604 9 2020-06-29 11:04:53.0 123 chunks SQL 0 bytes (0 avg bytes per chunk) Rows 45633 (371 avg rows per chunk)
1604 10 2020-06-29 11:04:53.0 123 chunks SQL 0 bytes (0 avg bytes per chunk) Rows 45633 (371 avg rows per chunk)
1604 11 2020-06-29 11:04:53.0 7 chunks SQL 0 bytes (0 avg bytes per chunk) Rows 2535 (362 avg rows per chunk)
   Event total: 1360 chunks 0 bytes in SQL statements 504498 rows
```

  Summary information is also output identicating an overall count of the changes. For example:

```
Total ROW chunks: 69487 with 18257671 updated rows (100%)
Total STATEMENT chunks: 0 with 0 bytes (0%)
628 events processed
```

  This information can be useful when viewing or monitoring the replication progress as it can help to indicate and identify the size of a specific transaction, particularly if the transaction is large. This can be particularly useful in combination with the `-first` [269] and/or `-last` [269].

- `-sizessummary`

  Outputs only the size summary information for the requested THL:

```
shell> thl list -sizessummary
Total ROW chunks: 69487 with 18257671 updated rows (100%)
Total STATEMENT chunks: 0 with 0 bytes (0%)
628 events processed
```

- `-specs`

  Shows the column specifications, such as identified type, length, and additional settings, when viewing events within row-based replication. This can be helpful when examining THL data in heterogeneous replication deployments.

  For example:

```
shell> thl list -low 5282 -specs
SEQ# = 5282 / FRAG# = 0 (last frag)
- TIME = 2020-01-30 05:46:26.0
- EPOCH# = 5278
- EVENTID = mysql-bin.000017:0000000000001117;0
- SOURCEID = host1
- METADATA = [mysql_server_id=1687011;dbms_type=mysql;is_metadata=true;»
   service=firstrep;shard=tungsten_firstrep;heartbeat=MASTER_ONLINE]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- SQL(0) =
 - ACTION = UPDATE
 - SCHEMA = tungsten_firstrep
 - TABLE = heartbeat
 - ROW# = 0
  - COL(index=1 name= type=4 [INTEGER] length=8 unsigned=false blob=false desc=null) = 1
  - COL(index=2 name= type=4 [INTEGER] length=8 unsigned=false blob=false desc=null) = 1416
  - COL(index=3 name= type=12 [VARCHAR] length=0 unsigned=false blob=false desc=null) = [B@65b60280
  - COL(index=4 name= type=93 [TIMESTAMP] length=0 unsigned=false blob=false desc=null) = 2020-01-30 05:46:26.0
  - COL(index=5 name= type=93 [TIMESTAMP] length=0 unsigned=false blob=false desc=null) = 2020-05-03 12:05:47.0
  - COL(index=6 name= type=4 [INTEGER] length=8 unsigned=false blob=false desc=null) = 1015
  - COL(index=7 name= type=4 [INTEGER] length=8 unsigned=false blob=false desc=null) = 0
  - COL(index=8 name= type=12 [VARCHAR] length=0 unsigned=false blob=false desc=null) = [B@105e55ab
  - KEY(index=1 name= type=4 [INTEGER] length=8 unsigned=false blob=false desc=null) = 1
```

  When identifying the different data types, the following effects should be noted:

  - `CHAR` and `VARCHAR` are identified as type 12, `VARCHAR`
  - `SET` is identified as an `INTEGER`
  - When the value is either `NULL` or `0` (Zero), date and time fields are shown as type 0, `NULL`
  - `ENUM` is identified as an `OTHER`
  - `BLOB` and `TEXT` are identified as type 2004, `BLOB`

- `-timezone`

  Specify the timezone to use when display date or time values. When not specified, times are displayed using UTC.

## 8.19.4. thl tail Command

> **Note**
>
> The thl tail command was introduced in verion 7.0.3

The thl tail command can be used to view a live stream of THL as it is either being generated by an extractor, or received on a replica from an upstream extractor.

On its own, the command will output the THL entry, as shown in the following example:

```
shell> thl tail
SEQ# = 13 / FRAG# = 0 (last frag)
- FILE = thl.data.0000000001
- TIME = 2023-02-02 14:23:27.0
- EPOCH# = 7
- EVENTID = mysql-bin.072822:0000000000034499;68
- SOURCEID = centos1
- METADATA = [mysql_server_id=1000;mysql_thread_id=68;dbms_type=mysql;tz_aware=true;is_metadata=true;service=alpha;shard=tungsten_alpha;heartbeat=NEW5]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [##charset = UTF-8, autocommit = 1, sql_auto_is_null = 0, foreign_key_checks = 1, unique_checks = 1, time_zone = 'SYSTEM', sql_mode = 'NO_ENGI
- SCHEMA = tungsten_alpha
- SQL(0) = UPDATE tungsten_alpha.heartbeat SET source_tstamp= '2023-02-02 15:23:27.845', salt= 5, name= 'NEW5'  WHERE id= 1
```

With the addition of the `-sql` switch, the command will output just the SQL statements, as shown in the following example:

```
shell> thl tail -sql
/* seqno: 11, fragno: 0 (last), epoch: 7, 2023-02-02 14:07:18.0, mysql-bin.072822:0000000000033589;65, centos1 */
/* SEQ# = 11 - SQL rendering of row change events is not supported */

/* seqno: 12, fragno: 0 (last), epoch: 7, 2023-02-02 14:07:57.0, mysql-bin.072822:0000000000034044;67, centos1 */
USE tungsten_alpha;
UPDATE tungsten_alpha.heartbeat SET source_tstamp= '2023-02-02 15:07:57.108', salt= 4, name= 'NEW5'  WHERE id= 1;
```

In both instances, output to the screen will continue until CTRL+C is pressed.

## 8.19.5. thl index Command

The index command to thl provides a list of all the available THL files and the sequence number range stored within each file:

```
shell> thl index
LogIndexEntry thl.data.0000000001(0:113)
LogIndexEntry thl.data.0000000002(114:278)
LogIndexEntry thl.data.0000000003(279:375)
LogIndexEntry thl.data.0000000004(376:472)
LogIndexEntry thl.data.0000000005(473:569)
LogIndexEntry thl.data.0000000006(570:941)
LogIndexEntry thl.data.0000000007(942:1494)
LogIndexEntry thl.data.0000000008(1495:1658)
LogIndexEntry thl.data.0000000009(1659:1755)
LogIndexEntry thl.data.0000000010(1756:1852)
LogIndexEntry thl.data.0000000011(1853:1949)
LogIndexEntry thl.data.0000000012(1950:2046)
LogIndexEntry thl.data.0000000013(2047:2563)
```

The optional argument `-no-checksum` [271] ignores the checksum information on events in the event that the checksum is corrupt.

## 8.19.6. thl purge Command

The purge command to the thl command deletes sequence number information from the THL files.

`thl` purge
[-low # ] | [-high # ]
[-y ] [-no-checksum ]

The purge command deletes the THL data according to the following rules:

- **Warning**

  > Purging all data requires that the THL information either be recreated from the source table, or reloaded from the Primary replicator.

  Without any specification, a purge command will delete all of the stored THL information.

- When only `-high` is specified, delete all the THL data up to and including the specified sequence number.
- When only `-low` is specified, delete all the THL data from and including the specified sequence number.
- With a range specification, using one or both of the `-low` and `-high` options, the range of sequences will be purged. The rules are the same as for the list command, enabling purge from the start to a sequence, from a sequence to the end, or all the sequences within a given range. The ranges must be on the boundary of one or more log files. It is not possible to delete THL data from the middle of a given file.

For example, consider the following list of THL files provided by thl index:

```
shell> thl index
LogIndexEntry thl.data.0000000377(5802:5821)
LogIndexEntry thl.data.0000000378(5822:5841)
LogIndexEntry thl.data.0000000379(5842:5861)
LogIndexEntry thl.data.0000000380(5862:5881)
LogIndexEntry thl.data.0000000381(5882:5901)
LogIndexEntry thl.data.0000000382(5902:5921)
LogIndexEntry thl.data.0000000383(5922:5941)
LogIndexEntry thl.data.0000000384(5942:5961)
LogIndexEntry thl.data.0000000385(5962:5981)
LogIndexEntry thl.data.0000000386(5982:6001)
LogIndexEntry thl.data.0000000387(6002:6021)
LogIndexEntry thl.data.0000000388(6022:6041)
LogIndexEntry thl.data.0000000389(6042:6061)
LogIndexEntry thl.data.0000000390(6062:6081)
LogIndexEntry thl.data.0000000391(6082:6101)
LogIndexEntry thl.data.0000000392(6102:6121)
LogIndexEntry thl.data.0000000393(6122:6141)
```

```
LogIndexEntry thl.data.0000000394(6142:6161)
LogIndexEntry thl.data.0000000395(6162:6181)
LogIndexEntry thl.data.0000000396(6182:6201)
LogIndexEntry thl.data.0000000397(6202:6221)
LogIndexEntry thl.data.0000000398(6222:6241)
LogIndexEntry thl.data.0000000399(6242:6261)
LogIndexEntry thl.data.0000000400(6262:6266)
```

The above shows a range of THL sequences from 5802 to 6266.

To delete all of the THL from the start of the list, sequence no 5802, to 6021 (inclusive), use the `-high` to specify the highest number to be removed (6021):

```
shell> thl purge -high 6021
WARNING: The purge command will break replication if you delete all
events or delete events that have not reached all slaves.
Are you sure you wish to delete these events [y/N]?
y
Deleting events where SEQ# <=6021
2020-02-10 16:31:36,235 [ - main] INFO  thl.THLManagerCtrl Transactions deleted
```

Running a thl index, sequence numbers from 6022 to 6266 are still available:

```
shell> thl index
LogIndexEntry thl.data.0000000388(6022:6041)
LogIndexEntry thl.data.0000000389(6042:6061)
LogIndexEntry thl.data.0000000390(6062:6081)
LogIndexEntry thl.data.0000000391(6082:6101)
LogIndexEntry thl.data.0000000392(6102:6121)
LogIndexEntry thl.data.0000000393(6122:6141)
LogIndexEntry thl.data.0000000394(6142:6161)
LogIndexEntry thl.data.0000000395(6162:6181)
LogIndexEntry thl.data.0000000396(6182:6201)
LogIndexEntry thl.data.0000000397(6202:6221)
LogIndexEntry thl.data.0000000398(6222:6241)
LogIndexEntry thl.data.0000000399(6242:6261)
LogIndexEntry thl.data.0000000400(6262:6266)
```

To delete the last two THL files, specify the sequence number at the start of the file, 6242 to the `-low` to specify the sequence number:

```
shell> thl purge -low 6242 -y
WARNING: The purge command will break replication if you delete all
events or delete events that have not reached all slaves.
Deleting events where SEQ# >= 6242
2020-02-10 16:40:42,463 [ - main] INFO  thl.THLManagerCtrl Transactions deleted
```

A thl index shows the sequence as removed:

```
shell> thl index
LogIndexEntry thl.data.0000000388(6022:6041)
LogIndexEntry thl.data.0000000389(6042:6061)
LogIndexEntry thl.data.0000000390(6062:6081)
LogIndexEntry thl.data.0000000391(6082:6101)
LogIndexEntry thl.data.0000000392(6102:6121)
LogIndexEntry thl.data.0000000393(6122:6141)
LogIndexEntry thl.data.0000000394(6142:6161)
LogIndexEntry thl.data.0000000395(6162:6181)
LogIndexEntry thl.data.0000000396(6182:6201)
LogIndexEntry thl.data.0000000397(6202:6221)
LogIndexEntry thl.data.0000000398(6222:6241)
```

The confirmation message can be bypassed by using the `-y` option, which implies that the operation should proceed without further confirmation.

The optional argument `-no-checksum` [271] ignores the checksum information on events in the event that the checksum is corrupt.

When purging, the THL files must be writeable; the replicator must either be offline or stopped when the purge operation is completed.

A purge operation may fail for the following reasons:

- Fatal error: The disk log is not writable and cannot be purged.

  The replicator is currently running and not in the `OFFLINE` [195] state. Use trepctl offline to release the write lock n the THL files.

- Fatal error: Deletion range invalid; must include one or both log end points: low seqno=0 high seqno=1000

  An invalid sequence number or range was provided. The purge operation will refuse to purge events that do not exist in the THL files and do not match a valid file boundary, i.e. the low figure must match the start of one file and the high the end of a file. Use thl index to determine the valid ranges.

## 8.19.7. thl info Command

The info command to thl command provides the current information about the THL, including the identified log directory, sequence number range, and the number of individual events with the available span. The lowest and highest THL file and sizes are also given. For example:

```
shell> thl info
log directory = /opt/continuent/thl/alpha/
log files = 41
logs size = 193.53 MB
min seq# = 0
max seq# = 228
events = 228
oldest file = thl.data.0000000001 (95.48 MB, 2019-12-18 11:53:00)
newest file = thl.data.0000000041 (0.98 MB, 2019-12-18 12:34:32)
```

The optional argument `-no-checksum` [271] ignores the checksum information on events in the event that the checksum is corrupt.

## 8.19.8. thl help Command

The help command to the thl command outputs the current help message text.

# 8.20. The trepctl Command

The trepctl command provides the main status and management interface to Tungsten Replicator. The trepctl command is responsible for:

- Putting the replicator online or offline
- Pause a specific stage within the replicator
- Performing backup and restore operations
- Skipping events in the THL in the event of an issue
- Getting status and active configuration information

The operation and control of the command is defined through a series of command-line options which specify general options, replicator wide commands, and service specific commands that provide status and control over specific services.

The trepctl command by default operates on the current host and configured service. For installations where there are multiple services and hosts in the deployment. Explicit selection of services and hosts is handled through the use of command-line options, for more information see Section 8.20.1, “trepctl Options”.

`trepctl`
backup [ `-backup agent` ] [ `-limit s` ] [ `-storage agent` ]
capabilities
check
clear
clients [ `-json` ]
error
flush [ `-limit s` ]
heartbeat [ `-name` ] [ `-tz s` ] [ `-host name` ]
kill [ `-y` ]
load
offline [ `-all-services` ]
offline-deferred [ `-at-event event` ] [ `-at-heartbeat [heartbeat]` ] [ `-at-seqno seqno` ] [ `-at-time YYYY-MM-DD_hh:mm:ss` ] [ `-immediate` ]
online [ `-all-services` ] [ `-base-seqno x` ] [ `-force` ] [ `-from-event event` ] [ `-no-checksum` ] [ `-provision [SCN]` ] [ `-skip-seqno seqdef` ] [ `-until-event event` ] [ `-until-heartbeat [name]` ] [ `-until-seqno seqno` ] [ `-until-time YYYY-MM-DD_hh:mm:ss` ]
pause [ `-stage stage-to-pause` ] [ `-time value-in-seconds` ]
perf [ `-c` ] [ `-r` ] [ `-port number` ]
properties [ `-filter name` ] [ `-values` ]
purge [ `-limit s` ] [ `-y` ]
qs [ `-c` ] [ `-r` ]
reset [ `-all` ] [ `-db` ] [ `-relay` ] [ `-thl` ] [ `-y` ]
restore
resume [ `-stage stage-to-resume` ] [ `-retry N` ] [ `-service name` ]
services [ `-c` ] [ `-full` ] [ `-json` ] [ `-r` ]
servicetable [ `-c` ] [ `-r` ]
setdynamic [ `-property` ] [ `-value` ]
setrole [ `-role master` | `slave` | `relay` | `thl-applier` | `thl-client` | `thl-server` ] [ `-uri` ]

shard [ `-delete shard` ] [ `-insert shard` ] [ `-list` ] [ `-update shard` ]
status [ `-c` ] [ `-json` ] [ `-name channel-assignments` | `services` | `shards` | `stages` | `stores` | `tasks` | `watches` ] [ `-r` ]
thl [ `-compression` ] [ `-encryption` ]
unload [ `-y` ] [ `-verbose` ]
version
wait [ `-applied seqno` ] [ `-limit s` ] [ `-state st` ]

For individual operations, trepctl uses a sub-command structure on the command-line that specifies which operation is to be performed. There are two classifications of commands, global commands, which operate across all replicator services, and service-specific commands that perform operations on a specific service and/or host. For information on the global commands available, see Section 8.20.2, "trepctl Global Commands". Information on individual commands can be found in Section 8.20.3, "trepctl Service Commands".

## 8.20.1. trepctl Options

Table 8.30. trepctl Command-line Options

| Option | Description |
|---|---|
| `error` | Will output full stack trace of the last error, if any. If no error, response will be empty. |
| `-host name` | Host name of the replicator |
| `-port number` | Port number of the replicator |
| `-retry N` | Number of times to retry the connection |
| `-service name` | Name of the replicator service |
| `-verbose` | Enable verbose messages for operations |

Global command-line options enable you to select specific hosts and services. If available, trepctl will read the active configuration to determining the host, service, and port information. If this is unavailable or inaccessible, the following rules are used to determine which host or service to operate upon:

- If no host is specified, then trepctl defaults to the host on which the command is being executed.
- If no service is specified:
  - If only one service has been configured, then trepctl defaults to showing information for the configured service.
  - If multiple services are configured, then trepctl returns an error, and requests a specific service be selected.

To use the global options:

- `-host`

  Specify the host for the operation. The replicator service must be running on the remote host for this operation to work.

- `-port`

  Specify the base TCP/IP port used for administration. The default is port 10000; port 10001 is also used. When using different ports, `port` and `port+1` is used, i.e. if port 4996 is specified, then port 4997 will be used as well. When multiple replicators are installed on the same host, different numbers may be used.

- `-service`

  The servicename to be used for the requested status or control operation. When multiple services have been configured, the servicename must be specified.

```
shell> trepctl status
Processing status command...
Operation failed: You must specify a service name with the -service flag
```

  Starting in 6.0.4, if multiple services are configured but not specified, then a list of available services is provided:

```
shell> trepctl status
Processing status command...
Operation failed: You must specify a service name with the -service flag because there is more than one
service available. The currently available status commands are:
trepctl -service north status
trepctl -service north_from_east status
trepctl -service north_from_west status
```

- `-verbose`

Turns on verbose reporting of the individual operations. This includes connectivity to the replicator service and individual operation steps. This can be useful when diagnosing an issue and identifying the location of a particular problem, such as timeouts when access a remote replicator.

- `-retry`

  Retry the request operation the specified number of times. The default is 10.

## 8.20.2. trepctl Global Commands

The trepctl command supports a number of commands that are global, or which work across the replicator regardless of the configuration or selection of individual services.

Table 8.31. trepctl Replicator Wide Commands

| Option | Description |
|---|---|
| `kill` | Shutdown the replication services immediately |
| `services` | List the configured replicator services |
| `servicetable` | List all the currently configures services in a tabular format |
| `thl` | Interact with dynamic THL options |
| `version` | Show the replicator version number and build |

These commands can be executed on the current or a specified host. Because these commands operate for replicators irrespective of the service configuration, selecting or specifying a service is note required.

### 8.20.2.1. trepctl kill Command

The trepctl kill command terminates the replicator without performing any cleanup of the replicator service, THL or sequence number information stored in the database. Using this option may cause problems when the replicator service is restarted.

`trepctl` kill [ `-y` ]

When executed, trepctl will ask for confirmation:

```
shell> trepctl kill
Do you really want to kill the replicator process? [yes/NO]
```

The default is no. To kill the service, ignoring the interactive check, use the `-y` option:

```
shell> trepctl kill -y
Sending kill command to replicator
Replicator appears to be stopped
```

### 8.20.2.2. trepctl services Command

The trepctl services command outputs a list of the current replicator services configured in the system and their key parameters such as latest sequence numbers, latency, and state.

`trepctl` services [ `-full` ] [ `-json` ]

For example:

```
shell> trepctl services
Processing services command...
NAME              VALUE
----              -----
appliedLastSeqno: 2541
appliedLatency  : 0.48
role            : master
serviceName     : alpha
serviceType     : local
started         : true
state           : ONLINE
Finished services command...
```

For more information on the fields displayed, see Section E.2, “Generated Field Reference”.

For a replicator with multiple services, the information is output for each configured service:

```
shell> trepctl services
```

```
Processing services command...
NAME              VALUE
----              -----
appliedLastSeqno: 44
appliedLatency  : 0.692
role            : master
serviceName     : alpha
serviceType     : local
started         : true
state           : ONLINE
NAME              VALUE
----              -----
appliedLastSeqno: 40
appliedLatency  : 0.57
role            : slave
serviceName     : beta
serviceType     : remote
started         : true
state           : ONLINE
NAME              VALUE
----              -----
appliedLastSeqno: 41
appliedLatency  : 0.06
role            : slave
serviceName     : gamma
serviceType     : remote
started         : true
state           : ONLINE
Finished services command...
```

The information can be reported in JSON format by using the `-json` option to the command:

```
shell> trepctl services -json
[
  {
    "serviceType" : "local",
    "appliedLatency" : "0.48",
    "serviceName" : "alpha",
    "appliedLastSeqno" : "2541",
    "started" : "true",
    "role" : "master",
    "state" : "ONLINE"
  }
]
```

The information is output as an array of objects, one object for each service identified.

If the `-full` option is added, the JSON output includes full details of the service, similar to that output by the trepctl status command, but for each configured service:

```
shell> trepctl services -json -full
[
  {
    "masterConnectUri" : "",
    "rmiPort" : "10000",
    "clusterName" : "default",
    "currentTimeMillis" : "1370256230198",
    "state" : "ONLINE",
    "maximumStoredSeqNo" : "2541",
    "minimumStoredSeqNo" : "0",
    "pendingErrorCode" : "NONE",
    "masterListenUri" : "thl://host1:2112/",
    "pendingErrorSeqno" : "-1",
    "pipelineSource" : "jdbc:mysql:thin://host1:3306/",
    "serviceName" : "alpha",
    "pendingErrorEventId" : "NONE",
    "appliedLatency" : "0.48",
    "transitioningTo" : "",
    "relativeLatency" : "245804.198",
    "role" : "master",
    "siteName" : "default",
    "pendingError" : "NONE",
    "uptimeSeconds" : "246023.627",
    "latestEpochNumber" : "2537",
    "extensions" : "",
    "dataServerHost" : "host1",
    "resourcePrecedence" : "99",
    "pendingExceptionMessage" : "NONE",
    "simpleServiceName" : "alpha",
    "sourceId" : "host1",
    "offlineRequests" : "NONE",
    "channels" : "1",
    "version" : "Tungsten Replicator 7.1.2 build 81",
```

```
      "seqnoType" : "java.lang.Long",
      "serviceType" : "local",
      "currentEventId" : "mysql-bin.000007:0000000000001033",
      "appliedLastEventId" : "mysql-bin.000007:0000000000001033;0",
      "timeInStateSeconds" : "245803.753",
      "appliedLastSeqno" : "2541",
      "started" : "true"
   }
]
```

Auto-refresh support added in 6.0.1. Starting with Tungsten Cluster 6.0.1, trepctl services supports the `-r` option to support auto-refresh.

For more information on the fields displayed, see Section E.2, “Generated Field Reference”.

### 8.20.2.3. trepctl servicetable Command

The trepctl servicetable command outputs a list of all the current services and current status information in a tabular format to make it easy to determine multi-service installations.

`trepctl` servicetable

For example:

```
shell> trepctl servicetable
Processing servicetable command...
Service              | Status                        | Role       | MasterConnectUri              |      SeqNo |    Latency
-------------------- | ----------------------------- | ---------- | ----------------------------- | ---------- | ----------
alpha                | ONLINE                        | slave      | thl://trfiltera:2112/         |        322 |       0.00
beta                 | ONLINE                        | slave      | thl://ubuntuheterosrc:2112/   |         12 |    4658.59
Finished servicetable command...
```

The command also supports the auto-refresh option, `-r`:

```
shell> trepctl servicetable -r 5
```

For more information on the fields displayed, see Section E.2, “Generated Field Reference”.

### 8.20.2.4. trepctl thl Command

The trepctl thl command is used to dynamically enable or disable on-disk encryption and compression of THL.

`trepctl` thl [ `-compression` ] [ `-encryption` ]

**`-compression`**

Pass `enable` or `disable` to the `-compression` option to enable or disable THL compression accordingly.

**`-encryption`**

Pass `enable` or `disable` to the `-encryption` option to enable or disable THL encryption accordingly.

Pass `genkey` to the `-encryption` option to generate a new encryption key.

### 8.20.2.5. trepctl version Command

The trepctl version command outputs the version number of the specified replicator service.

`trepctl` version

```
shell> trepctl version
Tungsten Replicator 7.1.2 build 81
```

The system can also be used to obtain remote version:

```
shell> trepctl -host host2 version
Tungsten Replicator 7.1.2 build 81
```

Version numbers consist of two parts, the main version number which denotes the product release, and the build number. Updates and fixes to a version may use updated build numbers as part of the same product release.

## 8.20.3. trepctl Service Commands

The trepctl service commands operate per-service, that is, when there are multiple services in a configuration, the service name on which the command operates must be explicitly stated. For example, when a backup is executed, the backup executes on an explicit, specified service.

The individuality of different services is critical when dealing with the replicator commands. Services can be placed into online or offline states independently of each other, since each service will be replicating information between different hosts and environments.

Table 8.32. trepctl Service Commands

| Option | Description |
|---|---|
| backup | Backup database |
| capabilities | List the configured replicator capabilities |
| check | Generate consistency check |
| clear | Clear one or all dynamic variables |
| clients | List clients connected to this replicator |
| flush | Synchronize transaction history log to database |
| heartbeat | Insert a heartbeat event with optional name |
| load | Load the replication service |
| offline | Set replicator to OFFLINE state |
| offline-deferred | Set replicator OFFLINE at a future point in the replication stream |
| online | Set Replicator to ONLINE with start and stop points |
| pause | Pause the replicator. Specify the stage using the -stage option and optional time using -time |
| perf | Print detailed performance information |
| properties | Display a list of all internal properties |
| purge | Purge non-Tungsten logins on database |
| qs | Print a simplified quick replicator status |
| reset | Deletes the replicator service |
| restore | Restore database on specified host |
| resume | Resume a paused replicator. Specify the stage using the -stage option. |
| setdynamic | Set dynamic properties |
| setrole | Set replicator role |
| shard | List, add, update, and delete shards |
| status | Print replicator status information |
| unload | Unload the replication service |
| wait | Wait for the replicator to reach a specific state, time or applied sequence number |

The following sections detail each command individually, with specific options, operations and information.

## 8.20.3.1. trepctl backup Command

The trepctl backup command performs a backup of the corresponding database for the selected service.

`trepctl` backup [ `-backup agent` ] [ `-limit s` ] [ `-storage agent` ]

Where:

Table 8.33. trepctl backup Command Options

| Option | Description |
|---|---|
| `-backup agent` [282] | Select the backup agent |
| `-limit s` [282] | The period to wait before returning after the backup request |
| `-storage agent` [282] | Select the storage agent |

Without specifying any options, the backup uses the default configured backup and storage system, and will wait indefinitely until the backup process has been completed:

```
shell> trepctl backup
Backup completed successfully; URI=storage://file-system/store-0000000002.properties
```

The return information gives the URI of the backup properties file. This information can be used when performing a restore operation as the source of the backup. See Section 8.20.3.19, "trepctl restore Command". Different backup solutions may require that the replicator be placed into the `OFFLINE` [195] state before the backup is performed.

A log of the backup operation will be stored in the replicator log directory, in a file corresponding to the backup tool used (e.g. `mysqldump.log`).

If multiple backup agents have been configured, the backup agent can be selected on the command-line:

```
shell> trepctl backup -backup mysqldump
```

If multiple storage agents have been configured, the storage agent can be selected using the `-storage` [282] option:

```
shell> trepctl backup -storage file
```

A backup will always be attempted, but the timeout to wait for the backup to be started during the command-line session can be specified using the `-limit` [282] option. The default is to wait indefinitely. However, in a scripted environment you may want to request the backup and continue performing other operations. The `-limit` [282] option specifies how long trepctl should wait before returning.

For example, to wait five seconds before returning:

```
shell> trepctl -service alpha backup -limit 5
Backup is pending; check log for status
```

The backup request has been received, but not completed within the allocated time limit. The command will return. Checking the logs shows the timeout:

```
...  management.OpenReplicatorManager Backup request timed out: seconds=5
```

Followed by the successful completion of the backup, indicated by the URI provided in the log showing where the backup file has been stored.

```
... backup.BackupTask Storing backup result...
... backup.FileSystemStorageAgent Allocated backup location: »
    uri =storage://file-system/store-0000000003.properties
... backup.FileSystemStorageAgent Stored backup storage file: »
    file=/opt/continuent/backups/store-0000000003-mysqldump_2013-07-15_18-14_11.sql.gz length=0
... backup.FileSystemStorageAgent Stored backup storage properties: »
    file=/opt/continuent/backups/store-0000000003.properties length=314
... backup.BackupTask Backup completed normally: »
    uri=storage://file-system/store-0000000003.propertiess
```

The URI can be used during a restore.

## 8.20.3.2. trepctl capabilities Command

The `capabilities` command outputs a list of the supported capabilities for this replicator instance.

`trepctl` capabilities

The information output will depend on the configuration and current role of the replicator service. Different services on the same host may have different capabilities. For example:

```
shell> trepctl capabilities
Replicator Capabilities
  Roles:             [master, slave]
  Replication Model: push
  Consistency Check: true
  Heartbeat:         true
  Flush:             true
```

The fields output are as follows:

- *`Roles`*

  Indicates whether the replicator can be a *`master`* or *`slave`*, or both.

- *`Replication Model`*

  The model used by the replication system. The default model for MySQL for example is push, where information is extracted from the binary log and pushed to Replicas that apply the transactions. The pull model is used for heterogeneous deployments.

- *`Consistency Check`*

Indicates whether the internal consistency check is supported. For more information see Section 8.20.3.3, "trepctl check Command".

- *Heartbeat*

  Indicates whether the heartbeat service is supported. For more information see Section 8.20.3.8, "trepctl heartbeat Command".

- *Flush*

  Indicates whether the trepctl flush operation is supported.

## 8.20.3.3. trepctl check Command

The `check` command operates by running a CRC check on the schema or table specified, creating a temporary table containing the check data and values during the process. The data collected during this process is then written to a consistency table within the replication configuration schema and is used to verify the table data consistency on the Primary and the Replica.

### Warning

Because the check operation is creating a temporary table containing a CRC of each row within the specified schema or specific table, the size of the temporary table created can be quite large as it consists of CRC and row count information for each row of each table (within the specified row limits). The configured directory used by MySQL for temporary table creation will need a suitable amount of space to hold the temporary data.

## 8.20.3.4. trepctl clear Command

The trepctl clear command deletes any dynamic properties configured within the replicator service.

`trepctl` clear

Dynamic properties include the current active role for the service. The dynamic information is stored internally within the replicator, and also stored within a properties file on disk so that the replicator can be restarted.

For example, the replicator role may be temporarily changed to receive information from a different host or to act as a Primary in place of a Replica. The replicator can be returned to the initial configuration for the service by clearing this dynamic property:

```
shell> trepctl clear
```

## 8.20.3.5. trepctl clients Command

Outputs a list of the that have been connected to the Primary service since it went online. If a Replica service goes offline or is stopped, it will still be reported by this command.

`trepctl` clients [ `-json` ]

Where:

### Table 8.34. trepctl clients Command Options

| Option | Description |
|---|---|
| `-json` [283] | Output the information as JSON |

The command outputs the list of clients and the management port on which they can be reached:

```
shell> trepctl clients
Processing clients command...
host4:10000
host2:10000
host3:10000
Finished clients command...
```

A JSON version of the output is available when using the `-json` [283] option:

```
shell> trepctl clients -json
[
{
"rmiPort": "10000",
"rmiHost": "host4"
},
{
"rmiPort": "10000",
"rmiHost": "host2"
```

```
},
{
"rmiPort": "10000",
"rmiHost": "host3"
}
]
```

The information is divided first by host, and then by the RMI management port.

## 8.20.3.6. trepctl error Command

The trepctl error command will output a full stack trace of the last occuring error, if any,

An empty response will be returned in the event of there being no error

`trepctl` error

```
shell> trepctl -service <serviceName> error

Event application failed: seqno=10 fragno=0 message=Table hr.regions not found in database. Unable to generate a valid statement.
com.continuent.tungsten.replicator.applier.ApplierException: Table hr.regions not found in database. Unable to generate a valid statement.
at com.continuent.tungsten.replicator.applier.JdbcApplier.getTableMetadata(JdbcApplier.java:582)
at com.continuent.tungsten.replicator.applier.JdbcApplier.fillColumnNames(JdbcApplier.java:494)
at com.continuent.tungsten.replicator.applier.JdbcApplier.getColumnInformation(JdbcApplier.java:1236)
at com.continuent.tungsten.replicator.applier.MySQLApplier.applyOneRowChangePrepared(MySQLApplier.java:418)
at com.continuent.tungsten.replicator.applier.JdbcApplier.applyRowChangeData(JdbcApplier.java:1460)
at com.continuent.tungsten.replicator.applier.JdbcApplier.apply(JdbcApplier.java:1576)
at com.continuent.tungsten.replicator.applier.ApplierWrapper.apply(ApplierWrapper.java:100)
at com.continuent.tungsten.replicator.pipeline.SingleThreadStageTask.apply(SingleThreadStageTask.java:871)
at com.continuent.tungsten.replicator.pipeline.SingleThreadStageTask.runTask(SingleThreadStageTask.java:601)
at com.continuent.tungsten.replicator.pipeline.SingleThreadStageTask.run(SingleThreadStageTask.java:185)
at java.base/java.lang.Thread.run(Thread.java:834)
```

## 8.20.3.7. trepctl flush Command

On a Primary, the trepctl flush command synchronizes the database with the transaction history log, flushing the in memory queue to the THL file on disk. The operation is not supported on a Replica.

`trepctl` flush [ `-limit s` ]

Internally, the operation works by inserting a heartbeat event into the queue, and then confirming when the heartbeat event has been committed to disk.

To flush the replicator:

```
shell> trepctl flush
Master log is synchronized with database at log sequence number: 3622
```

The flush operation is always initiated, and by default trepctl will wait until the operation completes. Using the `-limit` option, the amount of time the command-line waits before returning can be specified:

```
shell> trepctl flush -limit 1
```

## 8.20.3.8. trepctl heartbeat Command

Inserts a heartbeat into the replication stream, which can be used to identify replication points.

`trepctl` heartbeat [ `-name` ] [ `-tz s` ]

The heartbeat system is a way of inserting an identifiable event into the THL that is independent of the data being replicated. This can be useful when performing different operations on the data where specific checkpoints must be identified.

To insert a standard heartbeat:

```
shell> trepctl heartbeat
```

When performing specific operations, the heartbeat can be given an name:

```
shell> trepctl heartbeat -name dataload
```

Heartbeats insert a transaction into the THL using the transaction metadata and can be used to identify whether replication is operating between replicator hosts by checking that the sequence number has been replicated to the Replica. Because a new transaction is inserted, the sequence number is increased, and this can be used to identify if transactions are being replicated to the Replica without requiring changes to the database. To check replication using the heartbeat:

1. Check the current transaction sequence number on the Primary:

```
shell> trepctl status
Processing status command...
NAME                     VALUE
----                     -----
appliedLastEventId     : mysql-bin.000009:0000000000008998;0
appliedLastSeqno       : 3630
...
```

2. Insert a heartbeat event:

```
shell> trepctl heartbeat
```

3. Check the sequence number again:

```
shell> trepctl status
Processing status command...
NAME                     VALUE
----                     -----
appliedLastEventId     : mysql-bin.000009:0000000000009310;0
appliedLastSeqno       : 3631
```

4. Check that the sequence number on the Replica matches:

```
shell> trepctl status
Processing status command...
NAME                     VALUE
----                     -----
appliedLastEventId     : mysql-bin.000009:0000000000009310;0
appliedLastSeqno       : 3631
```

Heartbeats are given implied names, but can be created with explicit names that can be tracked during specific events and operations.

For example, when loading a specific set of data, the information may be loaded and then a backup executed on the Replica before enabling standard replication. This can be achieved by configuring the Replica to go offline when a specific heartbeat event is seen, loading the data on the Primary, inserting the heartbeat when the load has finished, and then performing the Replica backup:

1. On the Replica:

```
Replica shell> trepctl offline-deferred -at-heartbeat dataload
```

The trepctl offline-deferred configures the Replica to continue in the online state until the specified event, in this case the heartbeat, is received. The deferred state can be checked by looking at the status output, and the *`offlineRequests`* field:

```
Processing status command...
NAME                     VALUE
----                     -----
appliedLastEventId     : mysql-bin.000009:0000000000008271;0
appliedLastSeqno       : 3627
appliedLatency         : 0.704
...
offlineRequests        : Offline at heartbeat event: dataload
```

2. On the Primary:

```
Primary shell> mysql newdb < newdb.load
```

3. Once the data load has completed, insert the heartbeat on the Primary:

```
Primary shell> trepctl heartbeat -name dataload
```

The heartbeat will appear in the transaction history log after the data has been loaded and will identify the end of the load.

4. When the heartbeat is received, the Replica will go into the offline state. Now a backup can be created with all of the loaded data replicated from the Primary. Because the Replica is in the offline state, no further data or changes will be recorded on the Replica

This method of identifying specific events and points within the transaction history log can be used for a variety of different purposes where the point within the replication stream without relying on the arbitrary event or sequence number.

### 8.20.3.8.1. trepctl heartbeat Time Zone Handling

When the Replicator inserts a heartbeart there is an associated timezone. The default uses the Replicator host's timezone.

The `-tz` option to the trepctl heartbeat command may be used to force the use of a specific timezone.

For example, use GMT as the timezone when inserting a heartbeat:

```
shell> trepctl heartbeat -tz NONE
```

Use the Replicator host's timezone to insert the heartbeat:

```
shell> trepctl heartbeat -tz HOST
```

Use the given timezone to insert the heartbeat:

```
shell> trepctl heartbeat -tz {valid timezone id}
```

If the MySQL server timezone is different from the host timezone (which is strongly not recommended), then `-tz {valid timezone id}` should be used instead where `{valid timezone id}` should be the same as the MySQL server timezone.

### 8.20.3.8.2. trepctl heartbeat Internal Implementation

Internally, the heartbeat system operates through a tag added to the metadata of the THL entry and through a dedicated `heartbeat` table within the schema created for the replicator service. The table contains the sequence number, event ID, timestamp and heartbeat name. The heartbeat information is written into a special record within the transaction history log. A sample THL entry can be seen in the output below:

```
SEQ# = 3629 / FRAG# = 0 (last frag)
- TIME = 2013-07-19 12:14:57.0
- EPOCH# = 3614
- EVENTID = mysql-bin.000009:0000000000008681;0
- SOURCEID = host1
- METADATA = [mysql_server_id=1687011;dbms_type=mysql;is_metadata=true;service=alpha;
     shard=tungsten_alpha;heartbeat=dataload]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [##charset = UTF-8, autocommit = 1, sql_auto_is_null = 0,
    foreign_key_checks = 1, unique_checks = 1, sql_mode = 'IGNORE_SPACE',
    character_set_client = 33, collation_connection = 33, collation_server = 8]
- SCHEMA = tungsten_alpha
- SQL(0) = UPDATE tungsten_alpha.heartbeat SET source_tstamp= '2013-07-19 12:14:57',
  salt= 9, name= 'dataload'  WHERE id= 1
```

During replication, Replicas identify the heartbeat and record this information into their own `heartbeat` table. Because the heartbeat is recorded into the transaction history log, the specific sequence number of the transaction, and the event itself can be easily identified.

## 8.20.3.9. trepctl load Command

Load the replicator service.

`trepctl` load

Load the replicator service. The service name must be specified on the command-line, even when only one service is configured:

```
shell> trepctl load
Operation failed: You must specify a service name using -service
```

The service name can be specified using the `-service` option:

```
shell> trepctl -service alpha load
Service loaded successfully: name=alpha
```

## 8.20.3.10. trepctl offline Command

The trepctl offline command puts the replicator into the offline state, stopping replication.

`trepctl` offline [ `-all-services` ] [ `-immediate` ]

To put the replicator offline:

```
shell> trepctl offline
```

While offline:

- Transactions are not extracted from the source dataserver.
- Transactions are not applied to the destination dataserver.

Certain operations on the replicator, including updates to the operating system and dataserver should be performed while in the offline state.

By default, the replicator goes offline in deferred mode, allowing the current transactions being read from the binary log, or applied to the dataserver to complete, the sequence number table in the database is updated, and the replicator is placed offline, stopping replication.

To stop replication immediately, within the middle of an executing transaction, use the `-immediate` option:

```
shell> trepctl offline -immediate
```

## 8.20.3.11. trepctl offline-deferred Command

The trepctl offline-deferred sets a future sequence, event or heartbeat as the trigger to put the replicator in the offline state.

`trepctl` offline-deferred [ `-at-event event` ] [ `-at-heartbeat [heartbeat]` ] [ `-at-seqno seqno` ] [ `-at-time YYYY-MM-DD_hh:mm:ss` ]

Where:

### Table 8.35. trepctl offline-deferred Command Options

| Option | Description |
|---|---|
| `-at-event event` [287] | Go offline at the specified event |
| `-at-heartbeat [heartbeat]` [287] | Go offline when the specified heartbeat is identified |
| `-at-seqno seqno` [287] | Go offline at the specified sequence number |
| `-at-time YYYY-MM-DD_hh:mm:ss` [287] | Go offline at the specified time |

The trepctl offline-deferred command can be used to put the replicator into an offline state at some future point in the replication stream by identifying a specific trigger. The replicator must be online when the trepctl offline-deferred command is given; if the replicator is not online, the command is ignored.

The offline process performs a clean offline event, equivalent to executing trepctl offline. See Section 8.20.3.10, “trepctl offline Command”.

The supported triggers are:

- `-at-seqno` [287]

  Specifies a transaction sequence number (GTID) where the replication will be stopped. For example:

  ```
  shell> trepctl offline-deferred -at-seqno 3800
  ```

  The replicator goes into offline at the end of the matching transaction. In the above example, sequence 3800 would be applied to the dataserver, then the replicator goes offline.

- `-at-event` [287]

  Specifies the event where replication should stop:

  ```
  shell> trepctl offline-deferred -at-event 'mysql-bin.000009:0000000000088140;0'
  ```

  Because there is not a one-to-one relationship between global transaction IDs and events, the replicator will go offline at a transaction that has an event ID higher than the deferred event ID. If the event specification is located within the middle of a THL transaction, the entire transaction is applied.

- `-at-heartbeat` [287]

  Specifies the name of a specific heartbeat to look for when replication should be stopped.

- `-at-time` [287]

  Specifies a time (using the format YYYY-MM-DD_hh:mm:ss) at which replication should be stopped. The time must be specified in full (date and time to the second).

  ```
  shell> trepctl offline-deferred -at-time 2013-09-01_00:00:00
  ```

  The transaction being executed at the time specified completes, then the replicator goes offline.

If any specified deferred point has already been reached, then the replicator will go offline anyway. For example, if the current sequence number is 3800 and the deferred sequence number specified is 3700, then the replicator will go offline immediately just as if the trepctl offline command has been used.

When a trigger is reached, For example if a sequence number is given, that sequence will be applied and then the replicator will go offline.

The status of the pending trepctl offline-deferred setting can be identified within the status output within the *`offlineRequests`* field:

```
shell> trepctl status
...
offlineRequests        : Offline at sequence number: 3810
```

Multiple trepctl offline-deferred commands can be given for each corresponding trigger type. For example, below three different triggers have been specified, sequence number, time and heartbeat event, with the status showing each deferred event separated by a semicolon:

```
shell> trepctl status
...
offlineRequests        : Offline at heartbeat event: dataloaded;Offline at »
    sequence number: 3640;Offline at time: 2013-09-01 00:00:00 EDT
```

Offline deferred settings are cleared when the replicator is put into the offline state, either manually or automatically.

## 8.20.3.12. trepctl online Command

The trepctl online command puts the replicator into the online state. During the state change from offline to online various options can be used to control how the replicator goes back on line. For example, the replicator can be placed online, skipping one or more faulty transactions or disabling specific configurations.

`trepctl` online [ `-all-services` ] [ `-base-seqno x` ] [ `-force` ] [ `-from-event event` ] [ `-no-checksum` ] [ `-provision [SCN]` ] [ `-skip-seqno seqdef` ] [ `-until-event event` ] [ `-until-heartbeat [name]` ] [ `-until-seqno seqno` ] [ `-until-time YYYY-MM-DD_hh:mm:ss` ]

Where:

Table 8.36. trepctl online Command Options

| Option | Description |
|---|---|
| `-all-services` | Place online all available services |
| `-base-seqno x` | On a Primary, restart replication using the specified sequence number |
| `-force` | Force the online state |
| `-from-event event` | Start replication from the specified event |
| `-no-checksum` | Disable checksums for all events when going online |
| `-provision [SCN]` | Start provisioning using the parallel extractor |
| `-skip-seqno seqdef` | Skip one, multiple, or ranges of sequence numbers before going online |
| `-until-event event` | Define an event when replication will stop |
| `-until-heartbeat [name]` | Define a heartbeat when replication will stop |
| `-until-seqno seqno` | Define a sequence no when replication will stop |
| `-until-time YYYY-MM-DD_hh:mm:ss` | Define a time when replication will stop |

The trepctl online command attempts to switch replicator into the online state. The replicator may need to be put online because it has been placed offline for maintenance, or due to a failure.

To put the replicator online use the standard form of the command:

```
shell> trepctl online
```

Going online may fail if the reason for going offline was due to a fault in processing the THL, or in applying changes to the dataserver. The replicator will refuse to go online if there is a fault, but certain failures can be explicitly bypassed.

### 8.20.3.12.1. Going Online from Specific Transaction Points

If there is one, or more, event in the THL that could not be applied to the Replica because of a mismatch in the data (for example, a duplicate key), the event or events can be skipped using the `-skip-seqno` option. For example, the status shows that a statement failed:

```
shell> trepctl status
...
pendingError           : Event application failed: seqno=5250 fragno=0 »
    message=java.sql.SQLException: Statement failed on slave but succeeded on master
...
```

To skip the single sequence number, `5250`, shown:

```
shell> trepctl online -skip-seqno 5250
```

The sequence number specification can be specified according to the following rules:

- A single sequence number:

```
shell> trepctl online -skip-seqno 5250
```

- A sequence range:

```
shell> trepctl online -skip-seqno 5250-5260
```

- A comma-separated list of individual sequence numbers and/or ranges:

```
shell> trepctl online -skip-seqno 5250,5251,5253-5260
```

### 8.20.3.12.2. Going Online from a Base Sequence Number

#### Note

To set the position of the replicator, the dsctl command can also be used.

Alternatively, the base sequence number, the transaction ID where replication should start, can be specified explicitly:

```
shell> trepctl online -base-seqno 5260
```

#### Warning

Use of `-base-seqno` should be restricted to replicators in the *`master`* role only. Use on Replicas may lead to duplication or corruption of data.

#### Note

If issuing `-base-seqno` and `-from-event` together, you need to also issue the `-force` option for it to work, otherwise a warning will be displayed

### 8.20.3.12.3. Going Online from a Specific Event

#### Note

To set the position of the replicator, the dsctl command can also be used.

If the source event (for example, the MySQL binlog position) is known, this can be used as the reference point when going online and restarting replication:

```
shell> trepctl online -from-event 'mysql-bin.000011:0000000000002552;0'
```

When used, replication will start from the next event within the THL. The event ID provided must be valid. The event cannot be found in the THL, the operation will fail.

#### Note

If issuing `-base-seqno` and `-from-event` together, you need to also issue the `-force` option for it to work, otherwise a warning will be displayed

### 8.20.3.12.4. Going Online Until Specific Transaction Points

There are times when it is useful to be able to online until a specific point in time or in the replication stream. For example, when performing a bulk load parallel replication may be enabled, but only a single applier stream is required once the load has finished. The replicator can be configured to go online for a limited period, defined by transaction IDs, events, heartbeats, or a specific time.

The replicator must be in the offline state before the deferred online specifications are made. Multiple deferred online states can be specified in the same command when going online.

The setting of a future offline state can be seen by looking at the *`offlineRequests`* field when checking the status:

```
shell> trepctl status
...
minimumStoredSeqNo     : 0
offlineRequests        : Offline at sequence number: 5262;Offline at time: 2014-01-01 00:00:00 EST
pendingError           : NONE
...
```

If the replicator goes offline for any reason before the deferred offline state is reached, the deferred settings are lost.

#### 8.20.3.12.4.1. Going Online Until Specified Sequence Number

To go online until a specific transaction ID, use `-until-seqno`:

```
shell> trepctl online -until-seqno 5260
```

This will process all transactions up to, and including, sequence 5260, at which point the replicator will go offline.

#### 8.20.3.12.4.2. Going Online Until Specified Event

To go online until a specific event ID:

```
shell> trepctl online -until-event  'mysql-bin.000011:0000000000003057;0'
```

Replication will go offline when the event ID up to the specified event has been processed.

#### 8.20.3.12.4.3. Going Online Until Heartbeat

To go online until a heartbeat event:

```
shell> trepctl online -until-heartbeat
```

Heartbeats are inserted into the replication stream periodically, replication will stop once the heartbeat has been seen before the next transaction. A specific heartbeat can also be specified:

```
shell> trepctl online -until-heartbeat load-finished
```

#### 8.20.3.12.4.4. Going Online Until Specified Time

To go online until a specific date and time:

```
shell> trepctl online -until-time 2014-01-01_00:00:00
```

Replication will go offline once the transaction being processed at the time specified has completed.

### 8.20.3.12.5. Going Online by Force

In situations where the replicator needs to go online, the online state can be forced. This changes the replicator state to online, but provides no guarantees that the online state will remain in place if another, different, error stops replication.

```
shell> trepctl online -force
```

### 8.20.3.12.6. Going Online without Validating Checksum

In the event of a checksum problem in the THL, checksums can be disabled using the `-no-checksum` option:

```
shell> trepctl online -no-checksum
```

This will bring the replicator online without reading or writing checksum information.

> **Important**
>
> Use of the `-no-checksum` option disables both the reading and writing of checksums on log records. If starting the replicator without checksums to get past a checksum failure, the replicator should be taken offline again once the offending event has been replicated. This will avoid generating too many local records in the THL without checksums.

## 8.20.3.13. trepctl pause Command

The trepctl pause command allows you to pause a specific stage of the replicator, either indefinitely or for a specific time.

`trepctl` pause [ `-stage stage-to-pause` ] [ `-time value-in-seconds` ]

Where:

Table 8.37. trepctl pause Command Options

| Option | Description |
|---|---|
| `-stage stage-to-pause` | Used to specify the stage to pause, for example thl-to-q |
| `-time value-in-seconds` | Used to specify the time for a stage to remain paused. Specify in seconds. If not supplied, stage will pause indefinitely, or until a resume command is issued, or until the replicator is restarted. |

To pause the thl-to-q stage of the replicator use the standard form of the command:

```
shell> trepctl pause -stage thl-to-q
```

To pause the thl-to-q stage of the replicator for 2 minutes (120 seconds) use the command:

```
shell> trepctl pause -stage thl-to-q -time 120
```

If no time specified, the stage will remain paused until a resume command is issued, or until the replicator is restarted

## 8.20.3.14. trepctl perf Command

Display a list of all the internal properties. The list can be filtered.

`trepctl` perf [ `-c` ] [ `-r` ]

The perf outputs performance information on a stage by stage basis from the current replicator. The information has been reformatted and extracted from the existing replicator status, task and stage information available through other commands and requests, but reformatted and with values calculated to make identifying specific performance metrics quicker.

For example, on a typical extraction replicator:

```
Statistics since last put online 9265.385s ago
Stage       | Seqno | Latency |      Events |  Extraction |   Filtering |    Applying |       Other |       Total
binlog-to-q |  1604 |  8.779s |          14 |     60.173s |      0.109s |      0.015s |      0.004s |     60.301s
                        Avg time per Event |      4.298s |      0.008s |      0.000s |      0.001s |      4.307s
                          Filters in stage | colnames -> pkey
q-to-thl    |  1604 | 10.613s |          14 |     56.858s |      0.020s |      5.247s |      0.028s |     62.153s
                        Avg time per Event |      4.061s |      0.001s |      0.002s |      0.375s |      4.440s
                          Filters in stage | enumtostring -> settostring
```

On an applier:

```
Statistics since last put online 38.418s ago
Stage         | Seqno | Latency |      Events |  Extraction |   Filtering |    Applying |       Other |       Total
remote-to-thl |  3246 |  1.143s |          42 |     37.831s |      0.001s |      0.403s |      0.011s |     38.246s
                          Avg time per Event |      0.901s |      0.000s |      0.000s |      0.010s |      0.911s
thl-to-q      |  3246 |  1.209s |        1654 |     37.113s |      0.005s |      1.090s |      0.098s |     38.306s
                          Avg time per Event |      0.022s |      0.000s |      0.000s |      0.001s |      0.023s
q-to-dbms     |  3235 |  3.746s |        1644 |     22.226s |      0.019s |     15.242s |      0.338s |     37.825s
                          Avg time per Event |      0.014s |      0.000s |      0.000s |      0.009s |      0.023s
                            Filters in stage | mysqlsessions -> pkey
```

The individual statistics shown are as follows:

- All statistics within the replicator are reset when the replicator goes `ONLINE` [195]. The statistics shown are therefore displayed relative to the current uptime for the replicator.
- For each stage, the following information is shown:
  - Stage name
  - Seqno — this is the current *`SEQNO`* [549] number for the specified stage. A difference in sequence numbers is possible (as seen in the applier example above) during startup or synchronisation.
  - Latency — the latency of this stage compared to the commit time of the original transaction.
  - Events — the number of THL events processed by this stage.

  Statistics are then shown for each stage, two rows, first for the time to process all of the specified events, and then an average processing time for the events processed during that time within that stage. The individual statistics shown are as follows:

  - Extraction — the time taken to extract the event from the current source. On an extractor, this is the source database (for example, the binary log in MySQL). On other stages this is the time to read from disk or the remote replicator the THL event.
  - Filtering — the time taken to process the events through the filters configured in the specified stage.
  - Applying — the time taken to apply the event to the end of the stage, whether that is to THL on disk, the next queue in preparation for the next stage, or the target database.
  - Other — the time taken for other parts of the stage process, this includes waiting for thread management, updating internal structures, and recording information in the target datasource system, such as `trep_commit_seqno`.
- Filters in stage — The list of filters configured for this stage in the order in which they are applied to the event.

For convenience, the performance display can be set to refresh with a configured interval using the trepctl perf -r 5 command.

In the event that the replicator is currently offline, no statistics are displayed:

```
shell> trepctl perf
Currently not online; performance stats not available
State: Safely Offline for 6.491s
```

## 8.20.3.15. trepctl properties Command

Display a list of all the internal properties. The list can be filtered.

`trepctl` properties [ `-filter name` ] [ `-values` ]

The list of properties can be used to determine the current configuration:

```
shell> trepctl properties
{
"replicator.store.thl.log_file_retention": "7d",
"replicator.filter.bidiSlave.allowBidiUnsafe": "false",
"replicator.extractor.dbms.binlog_file_pattern": "mysql-bin",
"replicator.filter.pkey.url": »
    "jdbc:mysql:thin://host2:3306/tungsten_alpha?createDB=true",
...
}
```

### Note

Passwords are not displayed in the output.

The information is output as a JSON object with key/value pairs for each property and corresponding value.

The list can be filtered using the `-filter` option:

```
shell> trepctl properties -filter shard
{
"replicator.filter.shardfilter": »
    "com.continuent.tungsten.replicator.shard.ShardFilter",
"replicator.filter.shardbyseqno": »
    "com.continuent.tungsten.replicator.filter.JavaScriptFilter",
"replicator.filter.shardbyseqno.shards": "1000",
"replicator.filter.shardfilter.enforceHome": "false",
"replicator.filter.shardfilter.unknownShardPolicy": "error",
"replicator.filter.shardbyseqno.script": »
    "../../tungsten-replicator//samples/extensions/javascript/shardbyseqno.js",
"replicator.filter.shardbytable.script": »
    "../../tungsten-replicator//samples/extensions/javascript/shardbytable.js",
"replicator.filter.shardfilter.enabled": "true",
"replicator.filter.shardfilter.allowWhitelisted": "false",
"replicator.shard.default.db": "stringent",
"replicator.filter.shardbytable": »
    "com.continuent.tungsten.replicator.filter.JavaScriptFilter",
"replicator.filter.shardfilter.autoCreate": "false",
"replicator.filter.shardfilter.unwantedShardPolicy": "error"
}
```

The value or values from filtered properties can be retrieved by using the `-values` option:

```
shell>  trepctl properties -filter site.name -values
default
```

If a filter that would select multiple values is specified, all the values are listed without field names:

```
shell> trepctl properties -filter shard -values
com.continuent.tungsten.replicator.shard.ShardFilter
com.continuent.tungsten.replicator.filter.JavaScriptFilter
1000
false
../../tungsten-replicator//samples/extensions/javascript/shardbyseqno.js
error
../../tungsten-replicator//samples/extensions/javascript/shardbytable.js
true
false
stringent
com.continuent.tungsten.replicator.filter.JavaScriptFilter
false
error
```

## 8.20.3.16. trepctl purge Command

Forces all logins on the attached database, other than those directly related to Tungsten Cluster, to be disconnected. The command is only supported on a Primary, and can be used to disconnect users before a switchover or taking a Primary offline to prevent further use of the system.

`trepctl` purge [ `-limit s` ] [ `-y` ]

Where:

Table 8.38. trepctl purge Command Options

| Option | Description |
| --- | --- |
| `-limit s` [293] | Specify the waiting time for the operation |
| `-y` [293] | Indicates that the command should continue without interactive confirmation |

> **Warning**
>
> Use of the command will disconnect running users and queries and may leave the database is an unknown state. It should be used with care, and only when the dangers and potential results are understood.

To close the connections:

```
shell> trepctl purge
Do you really want to purge non-Tungsten DBMS sessions? [yes/NO]
```

You will be prompted to confirm the operation. To skip this confirmation and purge connections, use the `-y` [293] option:

```
shell> trepctl purge -y
Directing replicator to purge non-Tungsten sessions
Number of sessions purged: 0
```

An optional parameter, `-wait` [293], defines the period of time that the operation will wait before returning to the command-line.

An optional parameter, `-limit` [293], defines the period of time that the operation will wait before returning to the command-line.

## 8.20.3.17. trepctl qs Command

The trepctl qs (quickstatus) command provides a quicker, simpler, status display for the replicator showing only the critical information in a human-readable form. For example:

```
shell> trepctl qs
State: alpha Online for 4.21s, running for 1781.766s
Latency: 18.0s from source DB commit time on thl://ubuntuheterosrc.mcb:2112/ into target database
         1216.315s since last source commit
Sequence: 4804 last applied, 0 transactions behind (0-4804 stored) estimate 0.00s before synchronization
```

The information presented is as follows:

- State: alpha Online for 4.21s, running for 1781.766s

  The top line shows the basic status information about the replicator:

  - The name of the service (`alpha`).
  - The replicator's current state (`Online`) and the time in that state.
  - The amount of time the replicator has been running.

- Latency: 18.0s from source DB commit time on thl://ubuntuheterosrc.mcb:2112/ into target database

  The second lines shows the latency information, the information shown is based on the role of the replicator. The above line is shown on an applier, where the latency information shows the write delay into the target database, where the information is coming from, and applying to the target database. For a Primary (extractor) the information shown describes the latency from extraction into the THL files:

  ```
  State: alpha Online for 1699091.442s, running for 1699093.138s
  Latency: 0.113s from DB commit time on ubuntuheterosrc into THL
           1679.354s since last database commit
  Sequence: 4859 last applied, 0 transactions behind (0-4859 stored) estimate 0.00s before synchronization
  ```

- 1216.315s since last source commit

  The next line shows the interval since the last time there was a database commit. On a Primary (extractor) is the time between the last database commit to the binary log and the information being written to THL. On a Replica, it's the time between the last database commit *on the source database* and when the transaction was written to the target.

- Sequence: 4804 last applied, 0 transactions behind (0-4804 stored) estimate 0.00s before synchronization

The last line shows the sequence information:

- The last applied sequence number (to THL on a Primary, or to the target database on a Replica).
- The number of transactions behind the current stored transaction list. This is an indication on a Replica of how far behind in transactions (not latency) the Replica is from the Primary.
- The range of transactions currently stored (from minimum to maximum stored sequence number).
- An estimate of the how long it will take to apply the outstanding transactions. The calculation is made by determining the average rate transactions are being applied (either extraction or applying) against the number of outstanding transactions. It assumes all outstanding transactions are of an equal size. The actual THL transaction size is not taken into account. For information on THL sizes, try the thl list -sizes command.

If the replicator is offline due to being deliberately placed offline using trepctl offline then the basic information and status is shown:

```
shell> trepctl qs
State: Safely Offline for 352.775s
```

In the event of a replicator failure of some kind this will be reported in the output:

```
State: alpha Faulty (Offline) for 2.613s
Error Reason: SEQNO 4859 did not apply
    Error: CSV loading failed: schema=test table=msg CSV file=/opt/continuent/tmp/staging/alpha/staging0/test-msg-4859.csv
» message=Wrapped com.continuent.tungsten.replicator.ReplicatorException: OS command failed: command=cqlsh --keyspace=test
» --execute="copy stage_xxx_msg (tungsten_opcode,tungsten_seqno,tungsten_row_id,tungsten_commit_timestamp,id,msg) from
» '/opt/continuent/tmp/staging/alpha/staging0/test-msg-4859.csv' with NULL='NULL';" rc=1 stdout= stderr=Connection
» error: ('Unable to complete the operation against any hosts', {})
```

## 8.20.3.18. trepctl reset Command

The trepctl reset command resets an existing replicator service, performing the following operations:

- Deleting the local THL and relay directories
- Removes the Tungsten schema from the dataserver
- Removes any dynamic properties that have previously been set

The service name must be specified, using `-service`.

`trepctl` reset [ `-all` ] [ `-db` ] [ `-relay` ] [ `-thl` ] [ `-y` ]

Where:

### Table 8.39. trepctl reset Command Options

| Option | Description |
|---|---|
| `-all` [295] | Deletes the thl directory, relay logs directory and tungsten database for the service. |
| `-db` [295] | Deletes the tungsten_{service_name} database for the service |
| `-relay` [295] | Deletes the relay directory for the service |
| `-thl` [295] | Deletes the thl directory for the service |
| `-y` [294] | Indicates that the command should continue without interactive confirmation |

To reset a replication service, the replication service must be offline and the service name must be specified:

```
shell> trepctl offline
```

Execute the trepctl reset command:

```
shell> trepctl -service alpha reset
Do you really want to delete replication service alpha completely? [yes/NO]
```

You will be prompted to confirm the deletion. To ignore the interactive prompt, use the `-y` [294] option:

```
shell> trepctl -service alpha reset -y
```

Then put the replicator back online again:

```
shell> trepctl online
```

You can also reset only part of the overall service by including one of the following options:

- Reset all components of the service.
- Reset the THL. This is equivalent to running thl purge.
- Reset the relay log contents.
- Reset the database, including emptying the `trep_commit_seqno` and other control tables.

## 8.20.3.19. trepctl restore Command

Restores the database on a host from a previous backup.

`trepctl` capabilities

Once the restore has been completed, the node will remain in the `OFFLINE` [195] state. The datasource should be switched `ONLINE` [195] using trepctl:

```
shell> trepctl online
```

Any outstanding events from the Primary will be processed and applied to the Replica, which will catch up to the current Primary status over time.

## 8.20.3.20. trepctl resume Command

The trepctl resume command allows you to resume a specific stage of the replicator that has been paused using the `pause` option.

`trepctl` resume [ `-stage stage-to-resume` ]

Where:

### Table 8.40. trepctl resume Command Options

| Option | Description |
|---|---|
| `-stage stage-to-resume` | Used to specify the stage to resume, for example thl-to-q |

To resume the thl-to-q stage of the replicator use the standard form of the command:

```
shell> trepctl resume -stage thl-to-q
```

## 8.20.3.21. trepctl setdynamic Command

The trepctl setdynamic command allows you to change certain dynamic properties without the need to execute tpm update

`trepctl` setdynamic [ `-property` ] [ `-value` ]

Where:

### Table 8.41. trepctl setdynamic Command Options

| Option | Description |
|---|---|
| `-property` | Specify the property to change |
| `-value` | Specify the value of the specified -property |

To change a property, specify the property using the `-property` parameter.

> **Important**
>
> To change a property dynamically, the service must first be `OFFLINE`

```
shell> trepctl setdynamic -property <property>
```

The list of properties that can be dynamically changed are as follows:

- `replicator.autoRecoveryMaxAttempts` : This allows you to dynamically alter the behaior of the autoRecovery options. A value specificed greater than 0 will enable the autoRevovery. Set a value of 0 to disable.

The following example enables autoRecovery with the MaxAttempts value of 10

```
shell> trepctl -service beta setdynamic -property replicator.autoRecoveryMaxAttempts -value 10
```

The following example disables autoRecovery

```
shell> trepctl -service beta setdynamic -property replicator.autoRecoveryMaxAttempts -value 0
```

- `replicator.thl.protocol.client.serialization` : This allows you to dynamically alter THL transfer protocol for In-Flight compression.

  The following example sets the protocol to DEFLAT

  ```
  shell> trepctl setdynamic -property replicator.thl.protocol.client.serialization -value DEFLATE
  ```

  For more information on THL transfer protocol, see Section 7.17, "THL Encryption and Compression"

## 8.20.3.22. trepctl setrole Command

The trepctl setrole command changes the role of the replicator service. This command can be used to change a configured host between Replica and Primary roles, for example during switchover.

`trepctl` setrole [ `-role master` | `slave` | `relay` | `thl-applier` | `thl-client` | `thl-server` ] [ `-uri` ]

Where:

### Table 8.42. trepctl setrole Command Options

| Option | Description |
|---|---|
| `-role` | Replicator role |
| `-uri` [296] | URI of the Primary |

To change the role of a replicator, specify the role using the `-role` parameter. The replicator must be offline when the role change is issued:

```
shell> trepctl setrole -role master
```

When setting a Replica, the URI of the Primary can be optionally supplied:

```
shell> trepctl setrole -role slave -uri thl://host1:2112/
```

See Section 7.3, "Understanding Replicator Roles" for more details on each role.

## 8.20.3.23. trepctl shard Command

The trepctl shard command provides and interface to the replicator shard system definition system.

`trepctl` shard [ `-delete shard` ] [ `-insert shard` ] [ `-list` ] [ `-update shard` ]

Where:

### Table 8.43. trepctl shard Command Options

| Option | Description |
|---|---|
| `-delete shard` | Delete a shard definition |
| `-insert shard` | Add a new shard definition |
| `-list` | List configured shards |
| `-update shard` | Update a shard definition |

The replicator shard system is used during multi-site replication configurations to control where information is replicated.

### 8.20.3.23.1. Listing Current Shards

To obtain a list of the currently configured shards:

```
shell> trepctl shard -list
shard_id master critical
alpha    sales   true
```

The shard map information can also be captured and then edited to update existing configurations:

```
shell> trepctl shard -list>shard.map
```

### 8.20.3.23.2. Inserting a New Shard Configuration

To add a new shard map definition, either enter the information interactively:

```
shell> trepctl shard -insert
Reading from standard input
...
1 new shard inserted
```

Or import from a file:

```
shell> trepctl shard -insert  < shard.map
Reading from standard input
1 new shard inserted
```

### 8.20.3.23.3. Updating an Existing Shard Configuration

To update a definition:

```
shell> trepctl shard -update  < shard.map
Reading from standard input
1 shard updated
```

### 8.20.3.23.4. Deleting a Shard Configuration

To delete a single shard definition, specify the shard name:

```
shell> trepctl shard -delete alpha
```

## 8.20.3.24. trepctl status Command

The trepctl status command provides status information about the selected data service. The status information by default is a generic status report containing the key fields of status information. More detailed service information can be obtained by specifying the status name with the `-name` parameter.

The format of the command is:

`trepctl` status [ `-c` ] [ `-json` ] [ `-name channel-assignments` | `services` | `shards` | `stages` | `stores` | `tasks` | `watches` ] [ `-r` ]

Where:

Table 8.44. trepctl status Command Options

| Option | Description |
|---|---|
| `-c` | Used with the -r option to refresh the status display *c* number of times. |
| `-json` | Output the information in JSON format |
| `-name` | Select a specific group of status information |
| `-r` | Refresh the status display |

For example, to get the basic status information:

```
shell> trepctl status
Processing status command...
NAME                     VALUE
----                     -----
appliedLastEventId     : mysql-bin.000007:0000000000001353;0
appliedLastSeqno       : 2504
appliedLatency         : 0.53
channels               : 1
clusterName            : default
currentEventId         : mysql-bin.000007:0000000000001353
currentTimeMillis      : 1369233160014
dataServerHost         : host1
extensions             :
latestEpochNumber      : 2500
masterConnectUri       :
masterListenUri        : thl://host1:2112/
maximumStoredSeqNo     : 2504
minimumStoredSeqNo     : 0
offlineRequests        : NONE
```

```
pendingError           : NONE
pendingErrorCode       : NONE
pendingErrorEventId    : NONE
pendingErrorSeqno      : -1
pendingExceptionMessage: NONE
pipelineSource         : jdbc:mysql:thin://host1:3306/
relativeLatency        : 1875.013
resourcePrecedence     : 99
rmiPort                : 10000
role                   : master
seqnoType              : java.lang.Long
serviceName            : alpha
serviceType            : local
simpleServiceName      : alpha
siteName               : default
sourceId               : host1
state                  : ONLINE
timeInStateSeconds     : 1874.512
transitioningTo        :
uptimeSeconds          : 1877.823
version                : Tungsten Replicator 7.1.2 build 81
Finished status command...
```

For more information on the field information output, see Section E.2, “Generated Field Reference”.

The `-r #` can be used to automatically refresh the output at the specified interval. For example, trepctl status -r 5 will refresh the output every 5 seconds.

### 8.20.3.24.1. Getting Detailed Status

More detailed information about selected areas of the replicator status can be obtained by using the `-name` option.

#### 8.20.3.24.1.1. Detailed Status: Channel Assignments

When using a single threaded replicator service, the trepctl status -name channel-assignments will output an empty status. In parallel replication deployments, the trepctl status -name channel-assignments listing will output the list of schemas and their assigned channels within the configured channel quantity configuration. For example, in the output below, only two channels are shown, although five channels were configured for parallel apply:

```
shell> trepctl status -name channel-assignments
Processing status command (channel-assignments)...
NAME      VALUE
----      -----
channel : 0
shard_id: test
NAME      VALUE
----      -----
channel : 0
shard_id: tungsten_alpha
Finished status command (channel-assignments)...
```

#### 8.20.3.24.1.2. Detailed Status: Services

The trepctl status -name services status output shows a list of the currently configure internal services that are defined within the replicator.

```
shell> trepctl status -name services
Processing status command (services)...
NAME              VALUE
----              -----
accessFailures  : 0
active          : true
maxChannel      : -1
name            : channel-assignment
storeClass      : com.continuent.tungsten.replicator.channel.ChannelAssignmentService
totalAssignments: 0
Finished status command (services)...
```

#### 8.20.3.24.1.3. Detailed Status: Shards

The trepctl status -name shards status output lists the individual shards in operation, most useful when parallel apply has been configured within the replicator, showing a summary of last applied sequences and the corresponding binlog references.

In an environemnt not configured with parallel apply, the shards output will just show a single entry

#### 8.20.3.24.1.4. Detailed Status: Stages

The trepctl status -name stages status output lists the individual stages configured within the replicator, showing each stage, configuration, filters and other parameters applied at each replicator stage:

```
shell> trepctl status -name stages
Processing status command (stages)...
NAME                 VALUE
----                 -----
applier.class      : com.continuent.tungsten.replicator.thl.THLStoreApplier
applier.name       : thl-applier
blockCommitRowCount: 1
committedMinSeqno  : 15
extractor.class    : com.continuent.tungsten.replicator.thl.RemoteTHLExtractor
extractor.name     : thl-remote
name               : remote-to-thl
processedMinSeqno  : -1
taskCount          : 1
NAME                 VALUE
----                 -----
applier.class      : com.continuent.tungsten.replicator.thl.THLParallelQueueApplier
applier.name       : parallel-q-applier
blockCommitRowCount: 10
committedMinSeqno  : 15
extractor.class    : com.continuent.tungsten.replicator.thl.THLStoreExtractor
extractor.name     : thl-extractor
name               : thl-to-q
processedMinSeqno  : -1
taskCount          : 1
NAME                 VALUE
----                 -----
applier.class      : com.continuent.tungsten.replicator.applier.MySQLDrizzleApplier
applier.name       : dbms
blockCommitRowCount: 10
committedMinSeqno  : 15
extractor.class    : com.continuent.tungsten.replicator.thl.THLParallelQueueExtractor
extractor.name     : parallel-q-extractor
filter.0.class     : com.continuent.tungsten.replicator.filter.TimeDelayFilter
filter.0.name      : delay
filter.1.class     : com.continuent.tungsten.replicator.filter.MySQLSessionSupportFilter
filter.1.name      : mysqlsessions
filter.2.class     : com.continuent.tungsten.replicator.filter.PrimaryKeyFilter
filter.2.name      : pkey
name               : q-to-dbms
processedMinSeqno  : -1
taskCount          : 5
Finished status command (stages)...
```

### 8.20.3.24.1.5. Detailed Status: Stores

The trepctl status -name stores status output lists the individual internal stores used for replicating THL data. This includes both physical (on disk) THL storage and in-memory storage. This includes the sequence number, file size and retention information.

For example, the information shown below is taken from a Primary service, showing the stages, `binlog-to-q` which reads the information from the binary log, and the in-memory `q-to-thl` that writes the information to THL.

```
shell> trepctl status -name stages
Processing status command (stages)...
NAME                 VALUE
----                 -----
applier.class      : com.continuent.tungsten.replicator.storage.InMemoryQueueAdapter
applier.name       : queue
blockCommitRowCount: 1
committedMinSeqno  : 224
extractor.class    : com.continuent.tungsten.replicator.extractor.mysql.MySQLExtractor
extractor.name     : dbms
name               : binlog-to-q
processedMinSeqno  : 224
taskCount          : 1
NAME                 VALUE
----                 -----
applier.class      : com.continuent.tungsten.replicator.thl.THLStoreApplier
applier.name       : autoflush-thl-applier
blockCommitRowCount: 10
committedMinSeqno  : 224
extractor.class    : com.continuent.tungsten.replicator.storage.InMemoryQueueAdapter
extractor.name     : queue
name               : q-to-thl
processedMinSeqno  : 224
taskCount          : 1
Finished status command (stages)...
```

When running parallel replication, the output shows the store name, sequence number and status information for each parallel replication channel:

```
shell> trepctl status -name stores
Processing status command (stores)...
```

```
NAME                      VALUE
----                      -----
activeSeqno             : 15
doChecksum              : false
flushIntervalMillis     : 0
fsyncOnFlush            : false
logConnectionTimeout    : 28800
logDir                  : /opt/continuent/thl/alpha
logFileRetainMillis     : 604800000
logFileSize             : 100000000
maximumStoredSeqNo      : 16
minimumStoredSeqNo      : 0
name                    : thl
readOnly                : false
storeClass              : com.continuent.tungsten.replicator.thl.THL
timeoutMillis           : 2147483647
NAME                      VALUE
----                      -----
criticalPartition       : -1
discardCount            : 0
estimatedOfflineInterval: 0.0
eventCount              : 1
headSeqno               : 16
intervalGuard           : AtomicIntervalGuard (array is empty)
maxDelayInterval        : 60
maxOfflineInterval      : 5
maxSize                 : 10
name                    : parallel-queue
queues                  : 5
serializationCount      : 0
serialized              : false
stopRequested           : false
store.0                 : THLParallelReadTask task_id=0 thread_name=store-thl-0 »
                          hi_seqno=16 lo_seqno=16 read=1 accepted=1 discarded=0 events=0
store.1                 : THLParallelReadTask task_id=1 thread_name=store-thl-1 »
                          hi_seqno=16 lo_seqno=16 read=1 accepted=0 discarded=1 events=0
store.2                 : THLParallelReadTask task_id=2 thread_name=store-thl-2 »
                          hi_seqno=16 lo_seqno=16 read=1 accepted=0 discarded=1 events=0
store.3                 : THLParallelReadTask task_id=3 thread_name=store-thl-3 »
                          hi_seqno=16 lo_seqno=16 read=1 accepted=0 discarded=1 events=0
store.4                 : THLParallelReadTask task_id=4 thread_name=store-thl-4 »
                          hi_seqno=16 lo_seqno=16 read=1 accepted=0 discarded=1 events=0
storeClass              : com.continuent.tungsten.replicator.thl.THLParallelQueue
syncInterval            : 10000
Finished status command (stores)...
```

### 8.20.3.24.1.6. Detailed Status: Tasks

The trepctl status -name tasks command outputs the current list of active tasks within a given service, with one block for each stage within the replicator service.

```
shell> trepctl status -name tasks
Processing status command (tasks)...
NAME                    VALUE
----                    -----
appliedLastEventId    : mysql-bin.000038:0000000011253929;-1
appliedLastSeqno      : 1604
appliedLatency        : 8.779
applyTime             : 0.015
averageBlockSize      : 3.500
cancelled             : false
commits               : 4
currentBlockSize      : 0
currentLastEventId    : mysql-bin.000038:0000000011253929;-1
currentLastFragno     : 11
currentLastSeqno      : 1604
eventCount            : 14
extractTime           : 60.173
filterTime            : 0.109
lastCommittedBlockSize: 12
lastCommittedBlockTime: 59.145
otherTime             : 0.004
stage                 : binlog-to-q
state                 : extract
taskId                : 0
timeInCurrentEvent    : 8804.187
NAME                    VALUE
----                    -----
appliedLastEventId    : mysql-bin.000038:0000000011253929;-1
appliedLastSeqno      : 1604
appliedLatency        : 10.613
applyTime             : 5.247
averageBlockSize      : 2.800
```

```
cancelled             : false
commits               : 5
currentBlockSize      : 0
currentLastEventId    : mysql-bin.000038:0000000011253929;-1
currentLastFragno     : 11
currentLastSeqno      : 1604
eventCount            : 14
extractTime           : 56.858
filterTime            : 0.02
lastCommittedBlockSize: 12
lastCommittedBlockTime: 5.092
otherTime             : 0.028
stage                 : q-to-thl
state                 : extract
taskId                : 0
timeInCurrentEvent    : 8802.323
Finished status command (tasks)...
```

The list of tasks and information provided depends on the role of the host, the number of stages, and whether parallel apply is enabled.

#### 8.20.3.24.1.7. Detailed Status: Watches

The trepctl status -name watches command outputs the current list of tasks the replicator is waiting on before a specific action.

For example, if you issue trepctl offline-deferred -at-seqno x, the the output of watches will show the stages waiting on the specific seqno.

The following example show the use of offline-deferred and the subsequent resulting output from watches

```
shell> trepctl offline-deferred -at-seqno 234
shell> trepctl status -name watches
Processing status command (watches)...
NAME       VALUE
----       -----
action   : cancel tasks
cancelled: false
committed: false
done     : false
matched  : [[0:false]]
predicate: SeqnoWatchPredicate seqno=234
stage    : remote-to-thl
NAME       VALUE
----       -----
action   : cancel tasks
cancelled: false
committed: false
done     : false
matched  : [[0:false]]
predicate: SeqnoWatchPredicate seqno=234
stage    : thl-to-q
NAME       VALUE
----       -----
action   : cancel tasks
cancelled: false
committed: false
done     : false
matched  : [[0:false]]
predicate: SeqnoWatchPredicate seqno=234
stage    : q-to-dbms
Finished status command (watches)...
```

### 8.20.3.24.2. Getting JSON Formatted Status

Status information can also be requested in JSON format. The content of the information is identical, only the representation of the information is different, formatted in a JSON wrapper object, with one key/value pair for each field in the standard status output.

Examples of the JSON output for each status output are provided below. For more information on the fields displayed, see Section E.2, “Generated Field Reference”.

trepctl status JSON Output

```
{
"uptimeSeconds": "2128.682",
"masterListenUri": "thl://host1:2112/",
"clusterName": "default",
"pendingExceptionMessage": "NONE",
"appliedLastEventId": "mysql-bin.000007:0000000000001353;0",
"pendingError": "NONE",
"resourcePrecedence": "99",
"transitioningTo": "",
"offlineRequests": "NONE",
```

```
"state": "ONLINE",
"simpleServiceName": "alpha",
"extensions": "",
"pendingErrorEventId": "NONE",
"sourceId": "host1",
"serviceName": "alpha",
"version": "Tungsten Replicator 7.1.2 build 81",
"role": "master",
"currentTimeMillis": "1369233410874",
"masterConnectUri": "",
"rmiPort": "10000",
"siteName": "default",
"pendingErrorSeqno": "-1",
"appliedLatency": "0.53",
"pipelineSource": "jdbc:mysql:thin://host1:3306/",
"pendingErrorCode": "NONE",
"maximumStoredSeqNo": "2504",
"latestEpochNumber": "2500",
"channels": "1",
"appliedLastSeqno": "2504",
"serviceType": "local",
"seqnoType": "java.lang.Long",
"currentEventId": "mysql-bin.000007:0000000000001353",
"relativeLatency": "2125.873",
"minimumStoredSeqNo": "0",
"timeInStateSeconds": "2125.372",
"dataServerHost": "host1"
}
```

### 8.20.3.24.2.1. Detailed Status: Channel Assignments JSON Output

```
shell> trepctl status -name channel-assignments -json
[
   {
      "channel" : "0",
      "shard_id" : "cheffy"
   },
   {
      "channel" : "0",
      "shard_id" : "tungsten_alpha"
   }
]
```

### 8.20.3.24.2.2. Detailed Status: Services JSON Output

```
shell> trepctl status -name services -json
[
   {
      "totalAssignments" : "2",
      "accessFailures" : "0",
      "storeClass" : "com.continuent.tungsten.replicator.channel.ChannelAssignmentService",
      "name" : "channel-assignment",
      "maxChannel" : "0"
   }
]
```

### 8.20.3.24.2.3. Detailed Status: Shards JSON Output

```
shell> trepctl status -name shards -json
[
   {
      "stage" : "q-to-dbms",
      "appliedLastEventId" : "mysql-bin.000007:0000000007224342;0",
      "appliedLatency" : "63.099",
      "appliedLastSeqno" : "2514",
      "eventCount" : "16",
      "shardId" : "cheffy"
   }
]
```

### 8.20.3.24.2.4. Detailed Status: Stages JSON Output

```
shell> trepctl status -name stages -json
[
   {
      "applier.name" : "thl-applier",
      "applier.class" : "com.continuent.tungsten.replicator.thl.THLStoreApplier",
      "name" : "remote-to-thl",
      "extractor.name" : "thl-remote",
      "taskCount" : "1",
      "committedMinSeqno" : "2504",
      "blockCommitRowCount" : "1",
```

```
      "processedMinSeqno" : "-1",
      "extractor.class" : "com.continuent.tungsten.replicator.thl.RemoteTHLExtractor"
   },
   {
      "applier.name" : "parallel-q-applier",
      "applier.class" : "com.continuent.tungsten.replicator.storage.InMemoryQueueAdapter",
      "name" : "thl-to-q",
      "extractor.name" : "thl-extractor",
      "taskCount" : "1",
      "committedMinSeqno" : "2504",
      "blockCommitRowCount" : "10",
      "processedMinSeqno" : "-1",
      "extractor.class" : "com.continuent.tungsten.replicator.thl.THLStoreExtractor"
   },
   {
      "applier.name" : "dbms",
      "applier.class" : "com.continuent.tungsten.replicator.applier.MySQLDrizzleApplier",
      "filter.2.name" : "bidiSlave",
      "name" : "q-to-dbms",
      "extractor.name" : "parallel-q-extractor",
      "filter.1.name" : "pkey",
      "taskCount" : "1",
      "committedMinSeqno" : "2504",
      "filter.2.class" : "com.continuent.tungsten.replicator.filter.BidiRemoteSlaveFilter",
      "filter.1.class" : "com.continuent.tungsten.replicator.filter.PrimaryKeyFilter",
      "filter.0.class" : "com.continuent.tungsten.replicator.filter.MySQLSessionSupportFilter",
      "blockCommitRowCount" : "10",
      "filter.0.name" : "mysqlsessions",
      "processedMinSeqno" : "-1",
      "extractor.class" : "com.continuent.tungsten.replicator.storage.InMemoryQueueAdapter"
   }
]
```

### 8.20.3.24.2.5. Detailed Status: Stores JSON Output

```
shell> trepctl status -name stores -json
[
   {
      "logConnectionTimeout" : "28800",
      "doChecksum" : "false",
      "name" : "thl",
      "flushIntervalMillis" : "0",
      "logFileSize" : "100000000",
      "logDir" : "/opt/continuent/thl/alpha",
      "activeSeqno" : "2561",
      "readOnly" : "false",
      "timeoutMillis" : "2147483647",
      "storeClass" : "com.continuent.tungsten.replicator.thl.THL",
      "logFileRetainMillis" : "604800000",
      "maximumStoredSeqNo" : "2565",
      "minimumStoredSeqNo" : "2047",
      "fsyncOnFlush" : "false"
   },
   {
      "storeClass" : "com.continuent.tungsten.replicator.storage.InMemoryQueueStore",
      "maxSize" : "10",
      "storeSize" : "7",
      "name" : "parallel-queue",
      "eventCount" : "119"
   }
]
```

### 8.20.3.24.2.6. Detailed Status: Tasks JSON Output

```
shell> trepctl status -name tasks -json
[
   {
      "filterTime" : "0.0",
      "stage" : "remote-to-thl",
      "currentLastFragno" : "1",
      "taskId" : "0",
      "currentLastSeqno" : "2615",
      "state" : "extract",
      "extractTime" : "604.297",
      "applyTime" : "16.708",
      "averageBlockSize" : "0.982     ",
      "otherTime" : "0.017",
      "appliedLastEventId" : "mysql-bin.000007:0000000111424440;0",
      "appliedLatency" : "63.787",
      "currentLastEventId" : "mysql-bin.000007:0000000111424440;0",
      "eventCount" : "219",
      "appliedLastSeqno" : "2615",
      "cancelled" : "false"
```

```
  },
  {
     "filterTime" : "0.0",
     "stage" : "thl-to-q",
     "currentLastFragno" : "1",
     "taskId" : "0",
     "currentLastSeqno" : "2615",
     "state" : "extract",
     "extractTime" : "620.715",
     "applyTime" : "0.344",
     "averageBlockSize" : "1.904     ",
     "otherTime" : "0.006",
     "appliedLastEventId" : "mysql-bin.000007:0000000111424369;0",
     "appliedLatency" : "63.834",
     "currentLastEventId" : "mysql-bin.000007:0000000111424440;0",
     "eventCount" : "219",
     "appliedLastSeqno" : "2615",
     "cancelled" : "false"
  },
  {
     "filterTime" : "0.263",
     "stage" : "q-to-dbms",
     "currentLastFragno" : "1",
     "taskId" : "0",
     "currentLastSeqno" : "2614",
     "state" : "apply",
     "extractTime" : "533.471",
     "applyTime" : "61.618",
     "averageBlockSize" : "1.160     ",
     "otherTime" : "24.052",
     "appliedLastEventId" : "mysql-bin.000007:0000000110392640;0",
     "appliedLatency" : "63.178",
     "currentLastEventId" : "mysql-bin.000007:0000000110392711;0",
     "eventCount" : "217",
     "appliedLastSeqno" : "2614",
     "cancelled" : "false"
  }
]
```

#### 8.20.3.24.2.7. Detailed Status: Tasks JSON Output

```
shell> trepctl status -name watches -json
```

### 8.20.3.25. trepctl unload Command

Unload the replicator service.

`trepctl` unload [ `-y` ]

Unload the replicator service entirely. An interactive prompt is provided to confirm the shutdown:

```
shell> trepctl unload
Do you really want to unload replication service alpha? [yes/NO]
```

To disable the prompt, use the `-y` [304] option:

```
shell> trepctl unload -y
Service unloaded successfully: name=alpha
```

The name of the service unloaded is provided for confirmation.

### 8.20.3.26. trepctl wait Command

The trepctl wait command waits for the replicator to enter a specific state, or for a specific sequence number to be applied to the dataserver.

`trepctl` wait [ `-applied seqno` ] [ `-limit s` ] [ `-state st` ]

Where:

Table 8.45. trepctl wait Command Options

| Option | Description |
| --- | --- |
| `-applied seqno` [305] | Specify the sequence number to be waited for |
| `-limit s` [305] | Specify the number of seconds to wait for the operation to complete |
| `-state st` [305] | Specify a state to be waited for |

The command will wait for the specified occurrence, of either a change in the replicator status (i.e. `ONLINE` [195]), or for a specific sequence number to be applied. For example, to wait for the replicator to go into the `ONLINE` [195] state:

```
shell> trepctl wait -state ONLINE
```

This can be useful in scripts when the state maybe changed (for example during a backup or restore operation), allowing for an operation to take place once the requested state has been reached. Once reached, trepctl returns with exit status 0.

To wait a specific sequence number to be applied:

```
shell> trepctl wait -applied 2000
```

This can be useful when performing bulk loads where the sequence number where the bulk load completed is known, or when waiting for a specific sequence number from the Primary to be applied on the Replica. Unlike the offline-deferred operation, no change in the replicator is made. Instead, trepctl simply returns with exit status 0 when the sequence number has bee successfully applied.

If the optional `-limit` [305] option is used, then trepctl waits for the specified number of seconds for the request event to occur. For example, to wait for 10 seconds for the replicator to go online:

```
shell> trepctl wait -state ONLINE -limit 10
Wait timed out!
```

If the requested event does not take place before the specified time limit expires, then trepctl returns with the message 'Wait timed out!', and an exit status of 1.

# 8.21. The tmonitor Command

The tmonitor command was added in version 6.1.13

tmonitor is a tool for the management and testing of external Prometheus exporters (node and mysqld), and for the testing of internal exporters (Replicator).

The tmonitor command is located in the `$CONTINUENT_ROOT/tungsten/cluster-home/bin` directory.

### Note

> The tmonitor command will only be available in the `PATH` if the Tungsten software has been installed with the configuration option `profile-script` [417] included.

See the table below for a list of valid arguments:

Table 8.46. tmonitor Common Options

| Option | Description |
|---|---|
| `--connector`, `-C` | Specify the Tungsten Connector exporter (test and Tungsten Clustering only) |
| `--connectorPort` | Set the port to access the Connector exporter (default: 8093) |
| `--external`, `-e`, `-x` | Operate only upon External-Scope exporters |
| `--help`, `-h` | Display help text |
| `--internal`, `-i` | Operate only upon Internal-Scope exporters |
| `--manager`, `-M` | Specify the Tungsten Manager exporter (test and Tungsten Clustering only) |
| `--managerPort` | Set the port to access the Manager exporter (default: 8092) |
| `--mysql`, `--mysqld`, `-m` | Specify the MySQL exporter |
| `--node`, `-n` | Specify the node exporter |
| `--replicator`, `-R` | Specify the Tungsten Replicator exporter (test only) |
| `--replicatorPort` | Set the port to access the Replicator exporter (default: 8091) |
| `-t`, `-T` | Issue '-t test' to show Tungsten-specific metric lines. Issue '-T test' to show meetric help and headers |
| `--verbose`, `-v` | |

Exporters that require an external binary to function (i.e. node_exporter and mysqld_exporter) are considered to have an External Scope.

Exporters that do not require an external binary to function (i.e. the replicator is considered to have an Internal Scope.

By default, tmonitor {action} will act upon all available exporters.

If any exporter is specified on the CLI, then only those listed on the CLI will be acted upon.

The curl command must be available in the PATH for the `status` and `test` actions to function.

tmonitor status will use curl to test the exporter on `localhost`.

tmonitor test will use curl to fetch and print the metrics from one or more exporters on `localhost`.

tmonitor install will configure the specified exporter (or all external exporters if none is specified) to start at boot.

tmonitor remove will stop the specified exporter (or all external exporters if none is specified) from starting at boot.

Both the `install` and `remove` actions will attempt to auto-detect the boot sub-system. Currently, `init.d` and `systemd` are supported.

```
shell> tmonitor help
...
>>> Usage:

   tmonitor [args] {action}

   = Actions available for all exporter scopes:

      status - validate the service via curl (short output)
      test - validate the service via curl (full output)

   = Actions available for External-scope exporters only:

      start - launch the exporter process
      stop - kill the exporter process

      install - configure the exporter to start at boot
      remove - stop the exporter from starting at boot


>>> Arguments:

   [-h|--help]
   [-v|--verbose]

   [-i|--internal] Only act upon exporters with an internal scope
   [-e|-x|--external] Only act upon exporters with an external scope

   --internal and --external may not be specified together.

   = Internal Scope Exporters:

     [-C|--connector] Specify the Tungsten Connector exporter
     [-M|--manager] Specify the Tungsten Manager exporter
     [-R--replicator] Specify the Tungsten Replicator exporter

   = External Scope Exporters:

     [-m|--mysql|--mysqld] Specify the MySQL exporter
     [-n|--node] Specify the Node exporter
```

Example: View the status of all exporters:

```
shell> tmonitor status
MySQL exporter running ok
Node exporter running ok
Tungsten Replicator exporter running ok
All exporters running ok (Up: MySQL, Node, Tungsten Replicator)
```

Example: Start all exporters:

```
shell> tmonitor start
tungsten@db1-demo:/home/tungsten # tmonitor start
Node exporter started successfully on port 9100.
MySQL exporter started successfully on port 9104.

shell> tmonitor start
The Node exporter is already running
The MySQL exporter is already running
```

Example: Test all exporters:

```
shell> tmonitor test | wc -l
3097
```

```
shell> tmonitor test | grep '== '
== Metrics for the mysql exporter:
== Metrics for the node exporter:
== Metrics for the replicator exporter:

shell> tmonitor test | less
```

Example: Stop all exporters:

```
shell> tmonitor stop
All exporters stopped
```

Example: View the status of all exporters filtered by scope:

```
shell> tmonitor status -i
Tungsten Replicator exporter running ok
All internal exporters running ok (Up: Tungsten Replicator)

shell> tmonitor status -x
MySQL exporter running ok
Node exporter running ok
All external exporters running ok (Up: MySQL, Node)
```

Example: Install init.d boot scripts for all external exporters:

```
shell> tmonitor install
ERROR: You must be root to install or remove boot services.

Please be sure to run tmonitor as root via sudo, for example:

sudo /opt/continuent/tungsten/cluster-home/bin/tmonitor install


shell> sudo /opt/continuent/tungsten/cluster-home/bin/tmonitor install
node_exporter init.d boot script installed and activated

Use either `sudo service node_exporter start` or `tmonitor --node start` now to start the Node exporter.

mysqld_exporter init.d boot script installed and activated

Use either `sudo service mysqld_exporter start` or `tmonitor --mysql start` now to start the MySQL exporter.
```

Example: Install systemd boot scripts for all external exporters:

```
shell> tmonitor install
ERROR: You must be root to install or remove boot services.

Please be sure to run tmonitor as root via sudo, for example:

sudo /opt/continuent/tungsten/cluster-home/bin/tmonitor install


shell> sudo /opt/continuent/tungsten/cluster-home/bin/tmonitor install
Created symlink from /etc/systemd/system/multi-user.target.wants/node_exporter.service to /etc/systemd/system/node_exporter.service.
node_exporter systemd boot script installed and enabled

Use either `sudo systemctl start node_exporter` or `tmonitor --node start` now to start the Node exporter.

Created symlink from /etc/systemd/system/multi-user.target.wants/mysqld_exporter.service to /etc/systemd/system/mysqld_exporter.service.
mysqld_exporter systemd boot script installed and enabled

Use either `sudo systemctl start mysqld_exporter` or `tmonitor --mysql start` now to start the MySQL exporter.
```

Example: Remove init.d boot scripts for all external exporters:

```
shell> tmonitor remove
ERROR: You must be root to install or remove boot services.

Please be sure to run tmonitor as root via sudo, for example:

sudo /opt/continuent/tungsten/cluster-home/bin/tmonitor remove

shell> sudo /opt/continuent/tungsten/cluster-home/bin/tmonitor remove
node_exporter is still running, unable to remove. Please run either `tmonitor --node stop` or `sudo service node_exporter stop`, then retry this operation

shell> tmonitor stop
Node exporter stopped
MySQL exporter stopped

shell> sudo /opt/continuent/tungsten/cluster-home/bin/tmonitor remove
node_exporter init.d boot script de-activated and removed
mysqld_exporter init.d boot script de-activated and removed
```

Example: Remove systemd boot scripts for all external exporters:

```
shell> tmonitor stop
Node exporter stopped
MySQL exporter stopped

shell> tmonitor remove
ERROR: You must be root to install or remove boot services.

Please be sure to run tmonitor as root via sudo, for example:

sudo /opt/continuent/tungsten/cluster-home/bin/tmonitor remove

shell> sudo /opt/continuent/tungsten/cluster-home/bin/tmonitor remove
node_exporter systemd unit boot script disabled and removed
mysqld_exporter systemd unit boot script disabled and removed
```

# 8.22. The tpasswd Command

Table 8.47. tpasswd Common Options

| Option | Description |
|---|---|
| `--create`, `-c` | Creates a new user/password |
| `--delete`, `-d` | Delete a user/password combination |
| `-e`, `--encrypted.password` | Encrypt the password |
| `--file`, `-f` | Specify the location of the security.properties file |
| `-help`, `-h` | Display help text |
| `-p`, `--password.file.location` | Specify the password file location |
| `--target`, `-t` | Specify the target application |
| `-ts`, `--truststore.location` | |

# 8.23. The tprovision Script

> tprovision was previously known as tungsten_provision_slave and was renamed in v7.0.0
>
> For instructions for releases older than this, please refer to the documentation here

The tprovision script allows you to easily provision, or reprovision, a database server using information from a remote host.

`tprovision` [ `-r`, `--check-repl-channel` ] [ `-c`, `--create-master` ] [ `--flush-after-sleep` ] [ `--help`, `-h` ] [ `--method`, `-m` ] [ `--port`, `-p` ] [ `--seed` ] [ `--source`, `-s` ] [ `--threads`, `-t` ]

Where:

Table 8.48. tprovision Command-line Options

| Option | Description |
|---|---|
| `-r`, `--check-repl-channel` | Check that the number of replication channels when using parallel apply is the same for both source and target. On by default, set to 0 to disable check. |
| `-c`, `--create-master` | Use to flag to the script that the node being provisioned needs to be a Primary. Valid for Composite Active/Active only. Forces the provision of a failed Primary and will reset services. |
| `--flush-after-sleep` | How long we wait (in seconds) to get a full database write lock before final rsync pass. Valid and required only for rsync method |
| `--help`, `-h` | Show help text |
| `--method`, `-m` | Backup method to use. Valid methods are mysqldump, xtrabackup, rsync (v7.0.2+), and mysqlclone (v7.1.0+). |
| `--port`, `-p` | Port to use to connect to MySQL when using mysqldump, or ssh port when using xtrabackup. |
| `--seed` | Just do a single non-locking pass when using rsync. |

| Option | Description |
| --- | --- |
| `--source`, `-s` | Server to use as a source for the backup |
| `--threads`, `-t` | Number of parallel threads to use for xtrabackup. Increasing this number on large databases may improve backup speeds. If not supplied, default will be based on the default for the revision of xtrabackup in use. |

### Important

It is recommend to run this script in a utility such as screen in case the terminal gets disconnected.

For both `mysqldump` and `xtrabackup` methods, the script will perform a streaming backup from the source node to the target node.

The script will automatically put all replication services offline prior to beginning. If the services were online, the script will put them back online following a successful completion. All THL logs will be cleared prior to going online. The replicator will start replication from the position reflected on the source host.

Provisioning will fail from a Replica that is stopped, or if the Replica is not in the `ONLINE` [195] state.

Using xtrabackup

The script will run validation prior to starting to make sure the needed scripts are available. The provision process will run Xtrabackup on the source server and stream the contents to the server you are provisioning. After taking the backup, the script will prepare the directory and restart the MySQL server

Using mysqldump

The script will run mysqldump by default.

Using rsync

### Note

Provisioning using rsync was introduced in v7.0.2

The script will copy the source database to the target using rsync, using two passes. The first pass will simply copy the database live. This will produce an inconsistent database on the target but will have seeded the target database.

Pass 2 will quiesce the source database and do a final rsync. Because pass 1 had already seeded the data, pass 2 should be quick, minimizing the downtime on the source database.

Note that quiescing the database can take some time while we wait for in process transactions to complete. By default, we will wait 5 seconds when the source is a replica. When provisioning from a primary, there is no default and you must specify a time using the `--sleep-after-flush` option, in seconds, to wait. If the write lock cannot be obtained during this time, the script will abort before performing pass 2.

To only run the first pass, use the --seed option. This is useful to seed the target in advance and can be done multiple times to get the target seeded ahead of time to reduce downtime during a maintenance window.

Using rsync is not recommended when the source database is a primary, due to the database being quiesced and thus potentially causing downtime. The script normally will not allow you to do this, however you can override this check with the option `--i-am-sure` and force a provision from a primary database.

Using mysqlclone

### Note

Provisioning using mysqlclone was introduced in v7.1.0, and requires MySQL 8.0.17+

Beginning with mysql version 8.0.17, you can clone a database using mysqlclone. This requires the mysql_clone.so plugin on both the donor (source) and target. The script will attempt to install it if it is not installed already.

### Note

Note that mysqlclone ONLY supports InnoDB table types.

Compatibility

The script only works with MySQL at this time.

Logging

The script will log output to the `/opt/continuent/service_logs` directory.

Example

To reprovision the Replica db3 from another Replica, db2. Using xtrabackup

```
db3-shell> tprovision --source db2 --method xtrabackup
```

To reprovision the Replica db3 from another Replica, db2. Using the default, mysqldump

```
db3-shell> tprovision --source db2
```

To reprovision the PRimary db1 from the Primary, db4, in the remote cluster. Using Xtrabackup. This is only applicable to Composite Active/Active topologies

```
db1-shell> tprovision --source db4 -c --method xtrabackup
```

To reprovision the replica db3 from another replica, db2, using rsync and only seeding the database on db3

```
db3-shell> tprovision --source db2 --method rsync --seed
```

After seeding (above), run the rsync again. The amount of time the database is locked now should be much less since we've already seeded the database changes. We will wait 10 seconds to acquire the lock:

```
db3-shell> tprovision --source db2 --method rsync --sleep-after-flush 10
```

Reprovision db3 from db2 using mysqlclone (MySQL 8.0.17+)

```
db3-shell> tprovision --source db2 --method mysqlclone
```

# 8.24. The tungsten_get_mysql_datadir Script

The tungsten_get_mysql_datadir command will gather and display actual running values for the data directory (datadir) checked against multiple sources (running mysql value, mysqld default value and the tungsten configuration) and then resolve any sym-links to the physical destination.

`tungsten_get_mysql_datadir` [ `--debug`, `-d` ] [ `--flavor`, `-f` ] [ `--flavorful`, `-F` ] [ `--help`, `-h` ] [ `--verbose`, `-v` ]

Where:

Table 8.49. tungsten_get_mysql_datadir Command-line Options

| Option | Description |
|---|---|
| `--debug`, `-d` | |
| `--flavor`, `-f` | Also print out the MySQL server type as a tag in lower case (i.e. mysql, percona or mariadb) |
| `--flavorful`, `-F` | Also print out the MySQL server type (i.e. MySQL Community, Percona Server or MariaDB Server) |
| `--help`, `-h` | Show help text |
| `--verbose`, `-v` | |

The sudo command is used along with the which command to located the `mysqld` executable so as to gather the defaults.

# 8.25. The tungsten_get_ports Script

Table 8.50. tungsten_get_ports Options

| Option | Description |
|---|---|
| `--extra`, `-x` | Display the listening IP and source pair too |
| `--help`, `-h` | Show help text |
| `--internal`, `-i` | Display only ports used for INTERNAL purposes |
| `--list`, `-l` | Display the static database |
| `--showhost`, `-s` | Display the short hostname as the first field |

The tungsten_get_ports command will display the running Tungsten processes and the associated TCP ports that those processes are listening on.

Example:

```
shell> tungsten_get_ports
REPLICATOR[3678] 2112 THL
REPLICATOR[3678] 8091 Prometheus
REPLICATOR[3678] 8097 REST API
REPLICATOR[3678] 10000 RMI/JMX
REPLICATOR[3678] 10001 RMI/JMX
REPLICATOR[3678] 32000 Wrapper Liveness Checks
REPLICATOR[3678] 42679 INTERNAL RMI
MANAGER[4117] 7800 Group Communications
MANAGER[4117] 8090 REST API
MANAGER[4117] 8092 Prometheus
MANAGER[4117] 9997 RMI/JMX
MANAGER[4117] 11999 Router Gateway Port for Connector
MANAGER[4117] 12000 RMI/JMX
MANAGER[4117] 32001 Wrapper Liveness Checks
MANAGER[4117] 42957 INTERNAL RMI
MANAGER[4117] 46131 INTERNAL RMI
CONNECTOR[4551] 3100 RMI/JMX
CONNECTOR[4551] 3101 RMI/JMX
CONNECTOR[4551] 3306 MySQL R/W
CONNECTOR[4551] 3307 MySQL R/O
CONNECTOR[4551] 8093 Prometheus
CONNECTOR[4551] 8096 REST API
CONNECTOR[4551] 32002 Wrapper Liveness Checks
```

## 8.26. The tungsten_health_check Script

The tungsten_health_check may be used less frequently than Section 8.27, “The tungsten_monitor Script” to check the cluster against known best practices. It implements the Tungsten Script Interface as well as these additional options.

tungsten_health_check [ --dataservices ] [ --diagnostic-package ] [ --directory ] [ --email ] [ --force ] [ --from ] [ --help, -h ] [ --ignore ] [ --info, -i ] [ --json ] [ --lock-dir ] [ --lock-timeout ] [ --mail ] [ --net-ssh-option=key=value ] [ --notice, -n ] [ --show-differences ] [ --subject ] [ --test-failover ] [ --test-recover ] [ --test-switch ] [ --validate ] [ --verbose, -v ]

Where:

Table 8.51. tungsten_health_check Command-line Options

| Option | Description |
|---|---|
| --dataservices | This list of dataservices to monitoring to |
| --diagnostic-package | Create a diagnostic package if any issues are found |
| --directory | The $CONTINUENT_ROOT directory to use for running this command. It will default to the directory you use to run the script. |
| --email | Email address to send to when mailing any notifications |
| --force | Continue operation even if script validation fails |
| --from | The from address for sending messages |
| --help, -h | Show help text |
| --ignore | Ignore notices that use this key |
| --info, -i | Display info, notice, warning, and error messages |
| --json | Output all messages and the return code as a JSON object |
| --lock-dir | Directory to store log and lock files in |
| --lock-timeout | The number of minutes to sleep a notice after sending it |
| --mail | Path to the mail program to use for sending messages |
| --net-ssh-option=key=value | Provide custom SSH options to use for SSH communication to other hosts. |
| --notice, -n | Display notice, warning, and error messages |
| --show-differences | Show any differences in Tungsten configuration |
| --subject | Email subject line |
| --test-failover | Test failover for each managed dataservice |

| Option | Description |
|---|---|
| `--test-recover` | Test recover for each managed dataservice |
| `--test-switch` | Test the switch command for each managed dataservice |
| `--validate` | Only run script validation |
| `--verbose`, `-v` | Verbose |

Each time the tungsten_health_check runs, it will run a standard set of checks. Additional checks may be turned on using command line options.

- Check for errors using tpm validate
- Check that all servers in the dataservice are running the same version of Continuent Tungsten

The script can be run manually:

```
shell> tungsten_health_check
```

All messages will be sent to `/opt/continuent/share/tungsten_health_check/lastrun.log`.

Sending results via email

The tungsten_health_check is able to send you an email when problems are found. It is suggested that you run the script as root so it is able to use the mail program without warnings.

Alerts are cached to prevent them from being sent multiple times and flooding your inbox. You may pass `--reset` to clear out the cache or `--lock-timeout` to adjust the amount of time this cache is kept. The default is 3 hours.

```
shell> tungsten_health_check --from=you@yourcompany.com --to=group@yourcompany.com
```

Showing manual configuration file changes

The tpm validate command will fail if you have manually changed a configuration file. The file differences may be added if you include the `--show-differences` argument.

Testing Continuent Tungsten functionality

Continuent Tungsten includes a testing infrastructure that you can use at any time. By adding the `--test-switch`, `--test-failover` or `--test-recover` arguments to the command, we will test these operations on each database server.

> **Caution**
>
> This will have an impact on dataservice availability. Limit this operation to maintenance windows or times when you can experience managed outages.

Compatibility

The script only works with MySQL at this time.

## 8.27. The tungsten_monitor Script

The tungsten_monitor script provides a mechanism for monitoring the cluster state when monitoring tools like Nagios aren't available. It implements the Tungsten Script Interface as well as these additional options.

`tungsten_monitor` [ `--check-log` ] [ `--connector-timeout` ] [ `--dataservices` ] [ `--diagnostic-package` ] [ `--directory` ] [ `--disk` ] [ `--elb-script` ] [ `--email` ] [ `--force` ] [ `--help`, `-h` ] [ `--ignore` ] [ `--info`, `-i` ] [ `--json` ] [ `--latency` ] [ `--lock-dir` ] [ `--lock-timeout` ] [ `--mail` ] [ `--max-backup-age` ] [ `--net-ssh-option` ] [ `--notice`, `-n` ] [ `--reset` ] [ `--subject` ] [ `--validate` ] [ `--verbose`, `-v` ]

Where:

Table 8.52. tungsten_monitor Command-line Options

| Option | Description |
|---|---|
| `--check-log` | Email any lines in the log file that match the egrep expression. --check-log=tungsten-manager/log/tmsvc.log:OFFLINE |
| `--connector-timeout` | Number of seconds to wait for a connector response |
| `--dataservices` | This list of dataservices to monitoring to |
| `--diagnostic-package` | Create a diagnostic package if any issues are found |

| Option | Description |
|---|---|
| `--directory` | The $CONTINUENT_ROOT directory to use for running this command. It will default to the directory you use to run the script. |
| `--disk` | Display a warning if any disk usage is above this percentage |
| `--elb-script` | The xinetd script name that is responding to ELB liveness checks |
| `--email` | Email address to send to when mailing any notifications |
| `--force` | Continue operation even if script validation fails |
| `--help`, `-h` | Show help text |
| `--ignore` | Ignore notices that use this key |
| `--info`, `-i` | Display info, notice, warning, and error messages |
| `--json` | Output all messages and the return code as a JSON object |
| `--latency` | The maximum allowed latency for replicators |
| `--lock-dir` | Directory to store log and lock files in |
| `--lock-timeout` | The number of minutes to sleep a notice after sending it |
| `--mail` | Path to the mail program to use for sending messages |
| `--max-backup-age` | Maximum age in seconds of valid backups |
| `--net-ssh-option` | Provide custom SSH options to use for communication to other hosts. |
| `--notice`, `-n` | Display notice, warning, and error messages |
| `--reset` | Remove all entries from the lock directory |
| `--subject` | Email subject line |
| `--validate` | Only run script validation |
| `--verbose`, `-v` | Verbose |

General Operation

Each time the tungsten_monitor runs, it will run a standard set of checks. The set of checks will be determined automatically based on the current node configuration (for example, connector-timeout check will only run if the node has a connector installed). Additional checks may be turned on using command line options.

- Check that all Tungsten services for this host are running
- Check that all replication services and datasources are ONLINE
- Check that replication latency does not exceed a specified amount
- Check that the local connector is responsive
- Check disk usage

An example of adding it to crontab:

```
shell> crontab -l
10 * * * * /opt/continuent/tungsten/cluster-home/bin/tungsten_monitor >/dev/null 2>/dev/null
```

All messages will be sent to `/opt/continuent/share/tungsten_monitor/lastrun.log`.

Note that when all tungsten_monitor checks pass, the script will not print anything to the standard output.

Sending results via email

The tungsten_monitor is able to send you an email when problems are found. It is suggested that you run the script as root so it is able to use the mail program without warnings.

Alerts are cached to prevent them from being sent multiple times and flooding your inbox. You may pass `--reset` to clear out the cache or `--lock-timeout` to adjust the amount of time this cache is kept. The default is 3 hours.

```
shell> crontab -l
10 * * * * /opt/continuent/tungsten/cluster-home/bin/tungsten_monitor --from=you@yourcompany.com \
    --to=group@yourcompany.com >/dev/null 2>/dev/null
```

Monitoring log files

The tungsten_monitor can optionally monitor log files for certain keywords. This example will alert you to any lines in `trepsvc.log` that include OFFLINE.

shell> **`tungsten_monitor`** --check-log=tungsten-replicator/log/trepsvc.log:OFFLINE

Monitoring backup status

Knowing you have a recent backup is an important part any Tungsten deployment. The tungsten_monitor will look for the latest backup across all datasources and compare it to the value `--max-backup-age`. This example will let you know if a valid backup has not been taken in 3 days.

```
shell> tungsten_monitor --max-backup-age=259200
```

Compatibility

The script only works with MySQL at this time.

# 8.28. The tungsten_mysql_ssl_setup Script

### Note

This script was introduced in version 7.1.1.

The tungsten_mysql_ssl_setup command is a utility script that acts as a direct replacement for the mysql_ssl_rsa_setup command which is not included with either Percona Server or MariaDB. This command will be called by tpm cert gen mysqlcerts instead of mysql_ssl_rsa_setup.

# 8.29. The tungsten_prep_upgrade Script

The script was added in version 6.0.0

The tungsten_prep_upgrade command is a utility script which assists in the upgrade process from earlier v5 Composite Active/Active topologies to the newer Composite Composite Active/Active topology available from v6 onwards.

```
tungsten_prep_upgrade [--all|--service] {service} [--path {fullpath_to_replicator_dir}] [arguments]
```

Where:

Table 8.53. tungsten_prep_upgrade Command-line Options

| Option | Description |
|---|---|
| `--alldb` | Loop through all services on this node (based on existing tungsten_* schemas). |
| `--debug` | Debug mode is VERY chatty, avoid it unless you really need it. |
| `--dump`, `-d` | Backup tracking databases using mysqldump (default: tungsten_{service}) |
| `--dumpdb`, `-D` | Specify database to dump |
| `--force`, `-f` | Force the operation. |
| `--help`, `-h` | Show help text |
| `--host`, `-H` | Specify the database hostname or IP address |
| `--keep`, `--get`, `-g`, `-k` | Get the current tracking position schema for the specified service and save it as json to a text file |
| `--offline`, `-o` | Take the specific service to the offline-deferred state at-heartbeat (default: offline_for_upg) |
| `--offlinehb`, `-O` | Specify the name of the heartbeat for the offline-deferred operation |
| `--password`, `-w` | Specify the database password |
| `--path`, `-p` | Full path to the cross-site replicator directory (default: /opt/replicator) |
| `--port`, `-P` | Specify the database listener port (default: 13306) |
| `--restore`, `-r` | Load the tracking database backup for the specified service. |
| `--service`, `-s` | Specify the service name to act upon. |

| Option | Description |
|---|---|
| `--start` | Bring up the cross-site replicator process |
| `--stop` | Gracefully shut down the cross-site replicator process |
| `--targetdir`, `-T` | Specify directory target for dump |
| `--user`, `-u` | Specify the database user |
| `--verbose`, `-v` | Show verbose output |

If --{service|all} is not specified, tungsten_prep_upgrade will attempt to derive a list of one or more service names from trepctl services.

For operations that require MySQL access, tungsten_prep_upgrade will attempt to auto-locate the database user and password from tpm reverse.

Below are various examples:

- Take all replicator services offline automatically, or take a specific service offline:

```
shell> tungsten_prep_upgrade -o
~or~
shell> tungsten_prep_upgrade --service london --offline
shell> tungsten_prep_upgrade --service tokyo --offline
```

> **Note**
>
> To invoke the actual deferred offline set with `--offline`, use the below CLUSTER-specific trepctl command (i.e. from `/opt/continuent`, not `/opt/replicator`) on the Primary hosts within each cluster:
>
> ```
> shell> trepctl heartbeat -name offline_for_upg
> ```

- Get (keep) the current tracking position schema for the specified service and save it as json to a text file (default: ~/position-{service}-YYYYMMDDHHMMSS.txt)

```
shell> tungsten_prep_upgrade -g
~or~
shell> tungsten_prep_upgrade --service nyc --get
(NOTE: saves to ~/position-nyc-YYYYMMDDHHMMSS.txt)
shell> tungsten_prep_upgrade --service tokyo --get
(NOTE: saves to ~/position-tokyo-YYYYMMDDHHMMSS.txt)
```

- Gracefully shut down the cross-site replicator process:

```
shell> tungsten_prep_upgrade --stop
```

- Backup (dump) tracking databases using mysqldump (default: tungsten_{service})

```
shell> tungsten_prep_upgrade -d --alldb
~or~
shell> tungsten_prep_upgrade --service london --dump
shell> tungsten_prep_upgrade --service tokyo --dump
```

- Load (restore) the tracking database backup for the specified service. --all is unavailable with --restore.

```
shell> tungsten_prep_upgrade -s nyc -u tungsten -w secret -r
shell> tungsten_prep_upgrade -s tokyo -u tungsten -w secret -r
~or~
shell> tungsten_prep_upgrade --service nyc --user tungsten --password secret --restore
shell> tungsten_prep_upgrade --service tokyo --user tungsten --password secret --restore
```

> **Note**
>
> A restore may take place after the cross-site replicator is uninstalled, and so certain information is required on the command line (i.e. service, user and password)

# 8.30. The tungsten_provision_thl Command

The tungsten_provision_thl command can be used to generate the THL required to provision a database with information from a MySQL Primary to a Replica. Because of the way the tool works, the tool is most useful in heterogeneous deployments where the data must be formatted and processed by the replicator for effective loading into the target database.

The tool operates as follows:

1. A mysqldump of the current database is taken from the current Primary.

2. The generated SQL from mysqldump is then modified so that the data is loaded into tables using the `BLACKHOLE` engine type. These statements still generate information within the MySQL binary log, but do not create any data.
3. A sandbox MySQL server is started, using the MySQL Sandbox tool.
4. A duplicate replicator is started, pointing to the sandbox MySQL instance, but sharing the same THL port and THL directory.
5. The modified SQL from mysqldump is loaded, generating events in the binary log which are extracted by the sandbox replicator.

Because the sandbox replicator works on the same THL port as the standard Primary replicator, the Replicas will read the THL from the sandbox replicator. Also, because it uses the same THL directory, the THL will be written into additional THL files. It doesn't matter whether there are existing THL data files, the new THL will be appended into files in the same directory.

The tool has the following pre-requisites, in addition to the main Appendix B, *Prerequisites* for Tungsten Replicator:

- A tarball of the Tungsten Replicator must be available so that the duplicate replicator can be created. The full path to the file should be used.
- The MySQL Sandbox tool must have been installed. For more information, see MySQL Sandbox.

  Installing MySQL Sandbox requires the `ExtUtils::MakeMaker` and `Test::Simple` Perl modules. You may install these through CPAN or a package manager:

  ```
  shell> yum install -y perl-ExtUtils-MakeMaker perl-Test-Simple
  ```

  After those packages are available, you can proceed with building MySQL Sandbox and installing it. If you do not have sudo access, make sure that `~/MySQL-Sandbox-3.0.44/bin` is added to `$PATH`

  ```
  shell> cd ~
  shell> wget https://launchpad.net/mysql-sandbox/mysql-sandbox-3/mysql-sandbox-3/+download/MySQL-Sandbox-3.0.44.tar.gz
  shell> tar -xzf MySQL-Sandbox-3.0.44.tar.gz
  shell> cd MySQL-Sandbox-3.0.44
  shell> perl Makefile.PL
  shell> make
  shell> make test
  shell> sudo make install
  ```

- A tarball of a MySQL release must be available to create the sandbox MySQL environment. The release should match the installed version of MySQL. The full path to the file should be used.
- The replicator deployment should already be installed. The Primary should be `OFFLINE` [195], but the command can place the replicator offline automatically as part of the provisioning process.

Once these prerequisites have been met, the basic method of executing the command is to specify the location of the Tungsten Replicator tarball, MySQL tarball and the databases that you want to provision:

```
shell> tungsten_provision_thl \
      --tungsten-replicator-package=/home/tungsten/tungsten-replicator-7.1.2-81.tar.gz \
      --mysql-package=/home/tungsten/mysql-5.6.20-linux-glibc2.5-x86_64.tar.gz \
      --schemas=test
NOTE >>The THL has been provisioned to mysql-bin.000025:493 on host1:3306
```

The command reports the MySQL binary log point and host on which the THL has been provisioned. Put the Tungsten Replicator back online from the reported position:

```
shell> trepctl online -from-event 000025:493
```

The Tungsten Replicator will start extracting from that position and continue with any additional changes. Check all Replicas to be sure they are online. The Replicas services will process all extracted entries.

## 8.30.1. Provisioning from RDS

The tungsten_provision_thl script is designed to run from a replication Primary connected to a standard MySQL instance. The standard commands will not work if you are using RDS as a Primary.

The simplest method is to add the `--extract-from` [317] argument to your command. This will make the script compatible with RDS. The drawback is that we are not able to guarantee a consistent provisioning snapshot in RDS unless changes to the database are stopped. The script will monitor the binary log position during the provisioning process and alert you if there are changes. After the script completes, run trepctl online to resume extraction from the Primary at the current binary log position.

```
shell> tungsten_provision_thl \
      --extract-from=rds \
      --tungsten-replicator-package=/home/tungsten/tungsten-replicator-7.1.2-81.tar.gz \
      --mysql-package=/home/tungsten/mysql-5.6.20-linux-glibc2.5-x86_64.tar.gz \
      --schemas=test
```

If you aren't able to stop access to the database, the script can provision from an RDS Read Replica. Before running tungsten_provision_thl, replication to the replica must be stopped. This may be done by running CALL mysql.rds_stop_replication; in an RDS shell. Call tungsten_provision_thl with the `--extract-from` [317] and `--extract-from-host` [317] arguments. The script will read the correct Primary position based on the Replica replication position. After completion, resume extraction from the Primary using the standard procedure.

```
# Run `CALL mysql.rds_stop_replication();` on the RDS Read Replica
shell> tungsten_provision_thl \
      --extract-from=rds-read-replica \
      --extract-from-host=rds-host2 \
      --tungsten-replicator-package=/home/tungsten/tungsten-replicator-7.1.2-81.tar.gz \
      --mysql-package=/home/tungsten/mysql-5.6.20-linux-glibc2.5-x86_64.tar.gz \
      --schemas=test
NOTE >>The THL has been provisioned to mysql-bin.000025:493 on rds-host1:3306
# Run `CALL mysql.rds_start_replication();` on the RDS Read Replica
```

## 8.30.2. tungsten_provision_thl Reference

The format of the command is:

`tungsten_provision_thl` [ `--cleanup-on-failure` ] [ `--clear-logs` ] [ `--directory` ] [ `--extract-from` `mysql-native-slave` | `rds` | `rds-read-replica` | `tungsten-slave` ] [ `--extract-from-host` ] [ `--extract-from-port` ] [ `--help`, `-h` ] [ `--info`, `-i` ] [ `--java-file-encoding` ] [ `--json` ] [ `--mysql-package` ] [ `--net-ssh-option` ] [ `--notice`, `-n` ] [ `--offline` ] [ `--online` ] [ `--quiet`, `-q` ] [ `--sandbox-directory` ] [ `--sandbox-mysql-port` ] [ `--sandbox-password` ] [ `--sandbox-rmi-port` ] [ `--sandbox-user` ] [ `--schemas` ] [ `--service` ] [ `--tungsten-replicator-package` ] [ `--validate` ] [ `--verbose`, `-v` ]

Where:

– `--cleanup-on-failure` [317]

| | |
|---|---|
| Option | `--cleanup-on-failure` [317] |
| Description | Cleanup the sandbox installations when the provision process fails |
| Value Type | boolean |
| Default | false |

– `--clear-logs` [317]

| | |
|---|---|
| Option | `--clear-logs` [317] |
| Description | Delete all THL and relay logs for the service |
| Value Type | boolean |
| Default | false |

– `--directory` [317]

| | |
|---|---|
| Option | `--directory` [317] |
| Description | Use this installed Tungsten directory as the base for all operations |
| Value Type | string |

– `--extract-from` [317]

| | | |
|---|---|---|
| Option | `--extract-from` [317] | |
| Description | The type of server you are going to extract from | |
| Value Type | string | |
| Valid Values | mysql-native-slave | A MySQL native Replica with binary logging enabled |
| | rds | An Amazon RDS instance |
| | rds-read-replica | An Amazon RDS read replica instance |
| | tungsten-slave | An instance with Tungsten Cluster already installed with generated THL |

– `--extract-from-host` [317]

| | |
|---|---|
| Option | `--extract-from-host` [317] |
| Description | The hostname of a different MySQL server that will be used as the source for mysqldump |

| | |
|---|---|
| Value Type | string |

The hostname of a different MySQL server that will be used as the source for mysqldump. When given, the script will use `SHOW SLAVE STATUS` to determine the binary log position on the Primary server. You must run `STOP SLAVE` prior to executing tungsten_provision_thl.

– `--extract-from-port` [318]

| | |
|---|---|
| Option | `--extract-from-port` [318] |
| Description | The listening port of a different MySQL server that will be used as the source for mysqldump |
| Value Type | numeric |

– `--help` [318]

| | |
|---|---|
| Option | `--help` [318] |
| Aliases | `-h` [318] |
| Description | Display the help message |
| Value Type | string |

– `--info` [318]

| | |
|---|---|
| Option | `--info` [318] |
| Aliases | `-i` [318] |
| Description | Provide information-level messages |
| Value Type | string |

– `--java-file-encoding` [318]

| | |
|---|---|
| Option | `--java-file-encoding` [318] |
| Description | Java platform charset |
| Value Type | string |

– `--json` [318]

| | |
|---|---|
| Option | `--json` [318] |
| Description | Provide return code and logging messages as a JSON object after the script finishes |
| Value Type | string |

– `--mysql-package` [318]

| | |
|---|---|
| Option | `--mysql-package` [318] |
| Description | The location of a the MySQL tar.gz package |
| Value Type | string |

– `--net-ssh-option` [318]

| | |
|---|---|
| Option | `--net-ssh-option` [318] |
| Description | Sets additional options for SSH usage by the system, such as port numbers and passwords. |
| Value Type | string |
| Default | default |

Sets options for the `Net::SSH` Ruby module. This allows you to set explicit SSH options, such as changing the default network communication port, password, or other information. For example, using `--net-ssh-option=port=80` [318] will use port 80 for SSH communication in place of the default port 22.

For more information on the options, see http://net-ssh.github.com/ssh/v2/api/classes/Net/SSH.html#M000002.

– `--notice` [318]

| | |
|---|---|
| Option | `--notice` [318] |

| | |
|---|---|
| Aliases | -n [318] |
| Description | Provide notice-level messages |
| Value Type | string |

– --offline [319]

| | |
|---|---|
| Option | --offline [319] |
| Description | Put required replication services offline before processing |
| Value Type | boolean |
| Default | false |

– --online [319]

| | |
|---|---|
| Option | --online [319] |
| Description | Put required replication services online after successful processing |
| Value Type | boolean |
| Default | false |

– --quiet [319]

| | |
|---|---|
| Option | --quiet [319] |
| Aliases | -q [319] |
| Description | Execute with the minimum of output |
| Value Type | string |

– --sandbox-directory [319]

| | |
|---|---|
| Option | --sandbox-directory [319] |
| Description | The location to use for storing the temporary replicator and MySQL server |
| Value Type | string |

– --sandbox-mysql-port [319]

| | |
|---|---|
| Option | --sandbox-mysql-port [319] |
| Description | The listening port for the MySQL Sandbox |
| Value Type | string |
| Default | 3307 |

– --sandbox-password [319]

| | |
|---|---|
| Option | --sandbox-password [319] |
| Description | The password for the MySQL sandbox user |
| Value Type | string |
| Default | secret |

– --sandbox-rmi-port [319]

| | |
|---|---|
| Option | --sandbox-rmi-port [319] |
| Description | The listening port for the temporary Tungsten Replicator |
| Value Type | string |
| Default | 10002 |

– --sandbox-user [319]

| | |
|---|---|
| Option | --sandbox-user [319] |
| Description | The MySQL user to create and use in the MySQL Sandbox |

| | |
|---|---|
| Value Type | string |
| Default | tungsten |

– `--schemas` [320]

| | |
|---|---|
| Option | `--schemas` [320] |
| Description | The provision process will be limited to these schemas |
| Value Type | string |

– `--service` [320]

| | |
|---|---|
| Option | `--service` [320] |
| Description | Replication service to read information from |
| Value Type | string |
| Default | alpha |

– `--tungsten-replicator-package` [320]

| | |
|---|---|
| Option | `--tungsten-replicator-package` [320] |
| Description | The location of a fresh Tungsten Replicator tar.gz package |
| Value Type | string |

– `--validate` [320]

| | |
|---|---|
| Option | `--validate` [320] |
| Description | Run the script validation for the provided options and files |
| Value Type | boolean |
| Default | false |

– `--verbose` [320]

| | |
|---|---|
| Option | `--verbose` [320] |
| Aliases | `-v` [320] |
| Description | Provide verbose-level error messages |
| Value Type | string |

# 8.31. The tungsten_purge_thl Command

The tungsten_purge_thl command was added in version 7.0.0

The tungsten_purge_thl is a read-only command to assist with identifying the safe removal of THL based on the following rules:

- Gather the last applied seqno from all Replica nodes and take the lowest one
- Find the current THL file which contains that seqno, then locate the previous one
- construct a thl purge command to remove thl thru the last seqno in the prev file

A replicator service name name may be optionally specified as the last argument, for example:

```
shell> tungsten_purge_thl alpha
```

The tpm option `purge-thl` can be used to perform more advanced actions including the actual purge of the thl files.

Table 8.54. tungsten_purge_thl Options

| Option | Description |
|---|---|
| `--auto`, `-A` | Automatically execute any needed commands that it is possible to handle. Edits to the configuration are not possible at this time. |
| `--debug`, `-d` | Enable additional debug output. |

| Option | Description |
|---|---|
| --help, -h | Show help text |
| --i-am-sure | Bypass the "Are You Sure?" prompt when using --auto. |
| --info, -i | |
| --path, -p | Use to supply full path to replicator executables used for thl and trepctl if not in default location. |
| --quiet, -q | |
| --test, -t | |
| --thl | Use to supply full path to replicator executables used for thl if not in default location. Ignores --path if supplied |
| -d, --debug | Use to supply full path to replicator executables used for trepctl if not in default location. Ignores --path if supplied |
| --verbose, -v | |

# 8.32. The tungsten_read_master_events Script

The tungsten_read_master_events displays the raw contents of the Primary datasource for the given THL records. It implements the Tungsten Script Interface as well as these additional options.

tungsten_read_master_events [ --directory ] [ --force ] [ --help, -h ] [ --high ] [ --info, -i ] [ --json ] [ --low ] [ --net-ssh-option ] [ --notice, -n ] [ --service ] [ --source ] [ --validate ] [ --verbose, -v ]

Where:

Table 8.55. tungsten_read_master_events Command-line Options

| Option | Description |
|---|---|
| --directory | The $CONTINUENT_ROOT directory to use for running this command. It will default to the directory you use to run the script. |
| --force | Continue operation even if script validation fails |
| --help, -h | Show help text |
| --high | Display events ending with this sequence number |
| --info, -i | Display info, notice, warning, and error messages |
| --json | Output all messages and the return code as a JSON object |
| --low | Display events starting with this sequence number |
| --net-ssh-option | Provide custom SSH options to use for SSH communication to other hosts. |
| --notice, -n | Display notice, warning, and error messages |
| --service | Replication service to read information from |
| --source | Determine metadata for the --after, --low, --high statements from this host |
| --validate | Only run script validation |
| --verbose, -v | Show verbose information during processing |

Display all information after a specific sequence number

This may be used when you have had a Primary failover or would like to see everything that happened after a certain event. It will read the start position from the sequence number passed and allow you to see all events, even if they were not extracted by the replication service.

```
shell> tungsten_read_master_events --after=1792
```

If you provide the --source option, the script will SSH to the host in question and read its THL information.

Display information between two sequence numbers

This will show the raw Primary data between the two sequence numbers. It is inclusive so the information for the --low option will be included. This will only work if the sourceId for both sequence numbers is the same.

```
shell> tungsten_read_master_events --low=4582 --high=4725
```

Compatibility

The script only works with MySQL at this time.

# 8.33. The tungsten_send_diag Script

The tungsten_send_diag command is a utility script which assists in the upload of files to Continuent support.

tungsten_send_diag may be used in place of the Section 9.5.5, "tpm diag Command" to generate a diagnostic package.

tungsten_send_diag [ --args, -a ] [ --case, -c ] [ --cleanup ] [ --contentType ] [ --debug ] [ --diag, -d ] [ --email, -e ] [ --file, -f ] [ --help, -h ] [ --tpm, -t ] [ --verbose, -v ]

Where:

Table 8.56. tungsten_send_diag Command-line Options

| Option | Description |
|---|---|
| --args, -a | Specify arguments to be passed to tpm diag. Arguments must be passed in quotes. Requires the --diag option. |
| --case, -c | Specify the support case number |
| --cleanup | When specified, will cause the removal of the diagnostic archive file generated by the --diag argument. |
| --contentType | Specify the Content-Type for a file you are uploading |
| --debug | Debug mode is VERY chatty, avoid it unless you really need it. |
| --diag, -d | Automatically generate a tpm diag zip file and upload it |
| --email, -e | Email address to embed into the uploaded file name |
| --file, -f | File name to upload |
| --help, -h | Show help text |
| --tpm, -t | Full path to the tpm command you wish to use to execute a tpm diag |
| --verbose, -v | Show verbose output |

You must specify either --diag, --tpm, or --file, but not both. For example:

```
shell> tungsten_send_diag --diag -c 1234
```

To have tpm diag gather all nodes, add the --args '--all', for example:

```
shell> tungsten_send_diag --diag -c 1234 --args '--all'
```

You must specify either --email or --case, and you may provide both if you wish. For example:

```
shell> tungsten_send_diag -f example.zip -e you@yourdomain.com -c 1234
```

Using --tpm to specify one or more tpm commands implies the --diag option, you do not need to specify --diag if you use --tpm (or -t). For example:

```
shell> tungsten_send_diag -c 1234 -t /opt/replicator/tungsten/tools/tpm
```

You may generate multiple diags by specifying multiple tpm binaries with multiple arguments, i.e.:

```
shell> tungsten_send_diag -c 1234 -t /opt/continuent/tungsten/tools/tpm -t /opt/replicator/tungsten/tools/tpm
```

# 8.34. The tungsten_skip_seqno Script

The tungsten_skip_seqno allows events to be skipped based on filters, allowing the Tungsten Replicator to come back online with less manual intervention.

tungsten_skip_seqno [ --auto ] [ --debug, -d ] [ --filter, -f ] [ --filters ] [ --help, -h ] [ --info, -i ] [ --max, -m ] [ --path, -p ] [ --quiet, -q ] [ --service, -s ] [ --tpm ] [ --trepctl ] [ --verbose, -v ]

Where:

Table 8.57. tungsten_skip_seqno Command-line Options

| Option | Description |
|---|---|
| `--auto` | Automatically skip when seqno is an error state - ONLY USE THIS WHEN ASKED TO BY CONTINUENT SUPPORT |
| `--debug`, `-d` | Implies -v and -i also. Debug mode is VERY chatty, avoid it unless you really need it. |
| `--filter`, `-f` | Specify a single regex to match in the pendingExceptionMessage. If a filter is specified, then the new behavior will be to skip if the regex matches the pendingExceptionMessage, and do not not skip if there is no match. The regex will be matched against the pendingExceptionMessage like this: /yourRegexValue/gm Example, to match foreign key errors in a specific table: foreign key constraint fails.*?`yourSchemaName`.`yourTableName` |
| `--filters` | Specify a plain text file with one or more regexes, one per line, to match in the pendingExceptionMessage. |
| `--help`, `-h` | |
| `--info`, `-i` | Display additional information, but not fully verbose |
| `--max`, `-m` | Specify the maximum number of iterations |
| `--path`, `-p` | Specify the path to the trepctl executable if not installed in default location. |
| `--quiet`, `-q` | |
| `--service`, `-s` | Specify the {service_name} to act against if more than one service running |
| `--tpm` | Specify tpm executable name and path if changed from default. |
| `--trepctl` | Specify trepctl executable name and path if changed from default. |
| `--verbose`, `-v` | Display additional information (implies --info) |

General Operation

By default, the tungsten_skip_seqno command will:

- Gather a list of replicator service names using trepctl services | grep serviceName
- Start an inifinite loop
- Loop thru all services or use the service specified on the cli using the `--service` option
- Check the service status via trepctl -service {serviceName_here} status -json
- If the `pendingErrorSeqno` is not -1, then process the error state
- By default, if there is an error condition, a detailed message is displayed, and the user may skip the seqno interactively
- if the tungsten_skip_seqno command is called with `--auto` then the seqno with the error will be skipped automatically
- if the maximum number of loops has been reached (default: 100), the script will exit. Use `--max` to adjust this value
- Sleep for 3 seconds by default, then iterate

# 8.35. The tungsten_skip_all Command

This command was introduced in version 6.1.13.

The tungsten_skip_all command assists with skipping replicator errors that you deem safe to skip.

> **Warning**
>
> Blindly skipping replication errors without fully understanding the consequences could lead to data drift. This action should only be performed providing a full understanding of the error has been analysed and deemed to be safe to skip by yourself and/or you business.

The tungsten_skip_all command performs the following steps:

- Gather a list of replicator service names using trepctl services | grep serviceName.
- Starts an inifinite loop.
- Loops through all services, or uses the service specified on the cli.
- Checks the service status via trepctl -service {serviceName_here} status -json.
- If the `pendingErrorSeqno` is not -1, then processes the error state.
- By default, if there is an error condition, a detailed message is displayed, and the user may skip the seqno interactively.
- If the tungsten_skip_all command is called with `--auto` then the seqno with the error will be skipped automatically.
- If the maximum number of loops has been reached (default: 100), the script will exit.
- Sleeps for 3 seconds by default, then iterate

Table 8.58. tungsten_skip_all Options

| Option | Description |
|---|---|
| `--auto` | Automatically skip and seqno is an error state - ONLY USE THIS WHEN ASKED TO BY CONTINUENT SUPPORT |
| `--debug`, `-d` | Debug mode if very chatty. Avoid it unless you really need to. Implies --verbose and --info. |
| `--help`, `-h` | |
| `--info`, `-i` | Displays additional information, but not fully verbose. |
| `--max`, `-m` | Specify the maimum number of iterations. |
| `--path`, `-p` | Supply full path to trepctl and thl executables if NOT installed in default directory |
| `--quiet`, `-q` | |
| `--service`, `-s` | Supply service name to use - Required if more than one replication service running |
| `--trepctl` | Supply name of trepctl executable if different from default. |
| `--verbose`, `-v` | Displays additional information. Implies --info |

# 8.36. The undeployall Command

The undeployall command removes startup the startup and reboot scripts crteated by deployall, disabling automatic startup and shutdown of available services.

To use, the tool should be executed with superuser privileges, either directly using sudo, or by logging in as the superuser and running the command directly:

```
shell> sudo deployall
 Removing any system startup links for /etc/init.d/treplicator ...
   /etc/rc0.d/K80treplicator
   /etc/rc1.d/K80treplicator
   /etc/rc2.d/S80treplicator
   /etc/rc3.d/S80treplicator
   /etc/rc4.d/S80treplicator
   /etc/rc5.d/S80treplicator
   /etc/rc6.d/K80treplicator
```

To enable the scripts on the system, use deployall.

# Chapter 9. The tpm Deployment Command

tpm, or the Tungsten Package Manager, is a complete configuration, installation and deployment tool for Tungsten Replicator. It includes some utility commands to simplify those and other processes. In order to provide a stable system, all configuration changes must be completed using tpm. tpm makes use of ssh enabled communication and the sudo support as required by the Appendix B, *Prerequisites*.

tpm can operate in two different ways when performing a deployment:

- tpm staging configuration — a tpm configuration is created by defining the command-line arguments that define the deployment type, structure and any additional parameters. tpm then installs all the software on all the required hosts by using ssh to distribute Tungsten Cluster and the configuration, and optionally automatically starts the services on each host. tpm manages the entire deployment, configuration and upgrade procedure.
- tpm INI configuration — tpm uses an INI file to configure the service on the local host. The INI file must be create on each host that will run Tungsten Cluster. tpm only manages the services on the local host; in a multi-host deployment, upgrades, updates, and configuration must be handled separately on each host.

For a more detailed comparison of the two systems, see Section 9.1, "Comparing Staging and INI tpm Methods".

During the staging-based configuration, installation and deployment, the tpm tool works as follows:

- tpm creates a local configuration file that contains the basic configuration information required by tpm. This configuration declares the basic parameters, such as the list of hosts, topology requirements, username and password information. These parameters describe top-level information, which tpm translates into more detailed configuration according to the topology and other settings.
- Within staging-based configuration, each host is accessed (using ssh), and various checks are performed, for example, checking database configuration, whether certain system parameters match required limits, and that the environment is suitable for running Tungsten Replicator.
- During an installation or upgrade, tpm copies the current distribution to each remote host.
- The core configuration file is then used to translate a number of template files within the configuration of each component of the system into the configuration properties files used by Tunsten. The configuration information is shared on every configured host within the service; this ensures that in the event of a host failure, the configuration can be recovered.
- The components of Tungsten Replicator are then started (installation) or restarted according to the configuration options.

Where possible, these steps are conducted in parallel to speed up the process and limit the interruption to services and operations.

This method of operation ensures:

- Active configurations and properties are not updated until validation is completed. This prevents a running installation from being affected by an incompatible or potentially dangerous change to the configuration.
- Enables changes to be made to the staging configuration before the configuration is deployed.
- Services are not stopped/restarted unnecessarily.
- During an upgrade or update, the time required to reconfigure and restart is kept to a minimum.

Because of this safe approach to performing configuration, downtime is minimized, and the configuration is always based on files that are separate from, and independent of, the live configuration.

> **Important**
>
> tpm always creates the active configuration from the combination of the template files and parameters given to tpm. This means that changes to the underlying property files within the configuration are overwritten by tpm when the service is configured or updated.

In addition to the commands that tpm supports for the installation and configuration, the command also supports a number of other utility and information modes, for example, the fetch command retrieves existing configuration information to your staging, while query returns information about an active configuration.

Using tpm is divided up between the commands that define the operation the command will perform, which are covered in Section 9.5, "tpm Commands"; configuration options, which determine the parameters that configure individual services, which are detailed in Section 9.8, "tpm Configuration Options"; and the options that alter the way tpm operates, covered in Section 9.3, "tpm Staging Configuration".

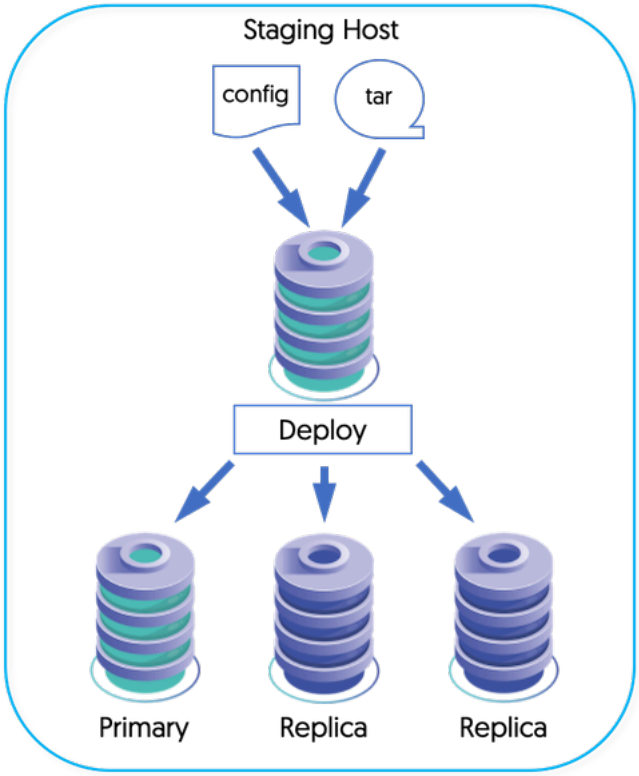
## 9.1. Comparing Staging and INI tpm Methods

tpm supports two different deployment methodologies. Both configure one or more Tungsten services, in a safe and secure manner, but differ in the steps and process used to complete the installation. The two methods are:

- Staging Directory

When using the staging directory method, a single configuration that defines all services and hosts within the deployment is created. tpm then communicates with all the hosts you are configuring to install and configure the different services required. This is best when you have a consistent configuration for all hosts and do not have any configuration management tools for your systems.

Figure 9.1. tpm Staging Based Deployment




- `INI` File

  When using the `INI` file method, configuration for each service must be made individually using an `INI` configuration file on each host. This is ideal for deployments where you have a configuration management system (e.g. Puppet and Chef) to manage the `INI` file. It also works very well for deployments where the configuration for each system is different from the others.

  Figure 9.2. tpm INI Based Deployment

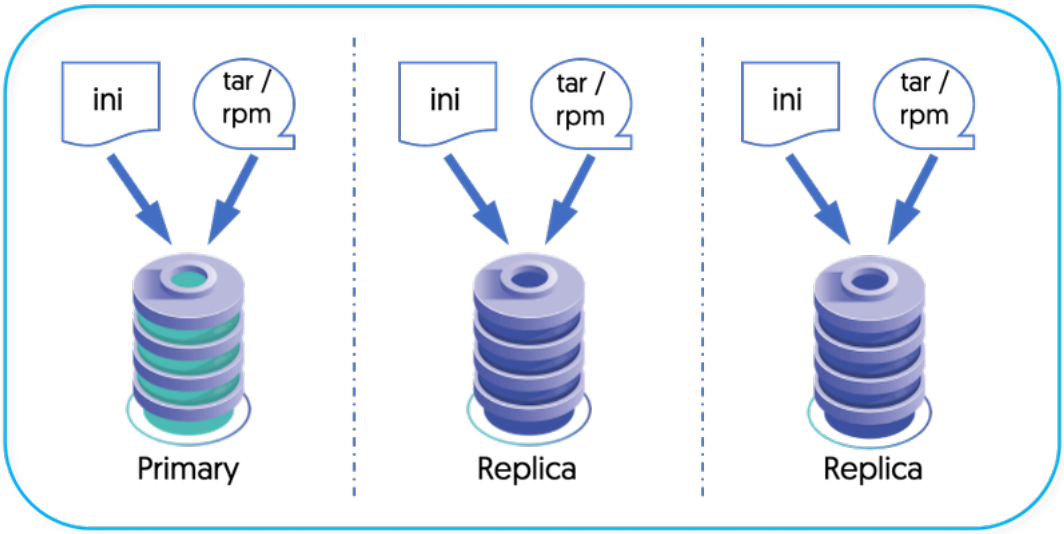


Table 9.1. TPM Deployment Methods

| Feature | Staging Directory | INI File |
|---|---|---|
| Deploy Multiple Services | Yes | Yes |
| Deploy to Multiple Hosts | Yes | No |
| Individual Host-based Configuration | Yes | Yes |
| Single-Step Upgrade | Yes | No |
| Requires SSH Configuration | Yes | No |
| RPM Support | Yes | Yes |

### Note

Check the output of tpm query staging to determine which method your current installation uses. The output for an installation from a staging directory will start with `# Installed from tungsten@staging-host:/opt/continuent/software/tungsten-replicator-7.1.2-81`. An installation based on an INI file may include this line but the hostname will reference the current host and there will be an `/etc/tungsten/tungsten.ini` file present.

To install a three-node service using the staging method:

1. Extract Tungsten Cluster on your staging server.
2. On each host:
   a. Complete all the Appendix B, *Prerequisites*, including setting the ssh keys.
3. Execute the tpm configure and tpm install commands to configure and deploy the service from the staging server.

To install a three-node service using the `INI` method:

1. On each host:
   a. Extract Tungsten Cluster.
   b. Complete all the Appendix B, *Prerequisites*.
   c. Create the `INI` file containing your configuration.
   d. Execute the tpm install command to deploy the service.

When using the staging method, upgrades and updates to the configuration must be made using tpm from the staging directory. Configuration methods can be swapped from staging to `INI` only by manually recreating the `INI` file with the new configuration and running tpm update.

## 9.2. Processing Installs and Upgrades

The tpm command is designed to coordinate the deployment activity across all hosts in a dataservice. This is done by completing a stage on all hosts before moving on. These operations will happen on each host in parallel and tpm will wait for the results to come back before moving on.

- Copy deployment files to each server

  At this stage, only the tpm command is copied over so we can run validation checks locally on each machine.

  The configuration is also transferred to each server and checked for completeness. This will run some commands to make sure that we have all of the settings needed to run a full validation.

- Validate the configuration settings

  Each host will validate the configuration based on validation classes. This will do things like check file permissions and database credentials. If errors are found during this stage, they will be summarized and the script will exit.

```
#######################################################################
# Validation failed
#######################################################################
#######################################################################
# Errors for host3
#######################################################################
```

```
ERROR >> host3 >> Password specified for app@% does not match the running instance on »
   tungsten@host3:13306 (WITH PASSWORD). This may indicate that the user has a password »
   using the old format. (MySQLConnectorPermissionsCheck)
######################################################################
# Errors for host2
######################################################################
ERROR >> host2 >> Password specified for app@% does not match the running instance on »
   tungsten@host2:13306 (WITH PASSWORD). This may indicate that the user has a password »
   using the old format. (MySQLConnectorPermissionsCheck)
######################################################################
# Errors for host1
######################################################################
ERROR >> host1 >> Password specified for app@% does not match the running instance on »
   tungsten@host1:13306 (WITH PASSWORD). This may indicate that the user has a password »
   using the old format. (MySQLConnectorPermissionsCheck)
```

At this point you should verify the configuration settings and retry the tpm install command. Any errors found during this stage may be skipped by running tpm configure alpha --skip-validation-check=MySQLConnectorPermissionsCheck. When re-running the tpm install command this check will be bypassed.

- Deploy and write configuration files

  If validation is successful, we will move on to deploying and writing the actual configuration files. The tpm command uses a JSON file that summarizes the configuration. The Tungsten processes use many different files to store the configuration and tpm is responsible for writing them.

  The `/opt/continuent/releases` directory will start to collect multiple directories after you have run multiple upgrades. We keep the previous versions in case a downgrade is needed or for review at a later date. If your upgrade has been successful, you can remove old directories. Make sure you do not remove the directory that is linked to by the `/opt/continuent/tungsten` symlink.

  > **Note**
  >
  > Do not change configuration files by hand. This will cause future updates to fail. One of the validation checks compares the file that tpm written with the current file. If there are differences, validation will fail.
  >
  > This is done to make sure that any configuration changes made by hand are not wiped out without giving you a chance to save them. You can run tpm query modified-files to see what, if any, changes have been made.

- Start Tungsten services

  After the installation is fully configured, the tpm command will start services on all of the hosts if the tpm option `--start` [424] was set. This process is slightly different depending on if you are doing a clean install or and upgrade.

  - Install

    1. Check if `--start` [424] or `--start-and-report` [424] were provided in the configuration
    2. Start Tungsten Replicator on all hosts

  - Upgrade

    1. Put all dataservices into *`MAINTENANCE`* mode
    2. Stop the Tungsten Replicator on all nodes

# 9.3. tpm Staging Configuration

Before installing your hosts, you must provide the desired configuration. This will be done with one or more calls to tpm configure as seen in Chapter 2, *Deployment Overview*. These calls place the given parameters into a staging configuration file that will be used during installation. This is done for dataservices, composite dataservices and replication services.

Instead of a subcommand, tpm configure accepts a service name or the word `defaults` as a subcommand. This identifies what you are configuring.

When configuring defaults, the defaults affect all configured services, with individual services able to override or set their own parameters.

```
shell> tpm configure [service_name|defaults] [tpm options] [service configuration options]
```

In addition to the Section 9.8, “tpm Configuration Options”, the common options in Table 9.18, “tpm Common Options” may be given.

The tpm command will store the staging configuration in the staging directory that you run it from. This behavior is changed if you have `$CONTINUENT_PROFILES` or `$REPLICATOR_PROFILES` defined in the environment. If present, tpm will store the staging configuration in that directory. Doing this will allow you to upgrade to a new version of the software without having to run the tpm fetch command.

If you are running Tungsten Replicator, the tpm command will use `$REPLICATOR_PROFILES` if it is available, before using `$CONTINUENT_PROFILES`.

## 9.3.1. Configuring default options for all services

```
shell> ./tools/tpm configure defaults \
    --replication-user=tungsten \
    --replication-password=secret \
    --replication-port=13306
```

These options will apply to all services in the configuration file. This is useful when working with a composite dataservice or multiple independent services. These options may be overridden by calls to tpm configure service_name or tpm configure service_name --hosts.

## 9.3.2. Configuring a single service

```
shell> ./tools/tpm configure alpha \
    --master=host1 \
    --members=host1,host2,host3 \
    --home-directory=/opt/continuent \
    --user=tungsten
```

The configuration options provided following the service name will be associated with the 'alpha' dataservice. These options will override any given with tpm configure defaults.

> Relationship of `--members` [413], `--slaves` [423] and `--master` [412]
>
> Each dataservice will use some combination of these options to define the hosts it is installed on. They define the relationship of servers for each dataservice.
>
> If you specify `--master` [412] and `--slaves` [423]; `--members` [413] will be calculated as the unique join of both values.
>
> If you specify `--master` [412] and `--members` [413]; `--slaves` [423] will be calculated as the unique difference of both values.

## 9.3.3. Configuring a single host

```
shell> ./tools/tpm configure alpha \
    --hosts=host3 \
    --backup-method=xtrabackup-incremental
```

This will apply the `--repl-backup-method` [396] option to just the host3 server. Multiple hosts may be given as a comma-separated list. The names used in the `--members` [413], `--slaves` [423], `--master` [412], options should be used when calling `--hosts` [409]. These values will override any given in tpm configure defaults or tpm configure alpha.

## 9.3.4. Reviewing the current configuration

You may run the tpm reverse command to review the list of configuration options. This will run in the staging directory and in your installation directory. It is a good idea to run this command prior to installation and upgrades to validate the current settings.

```
# Installed from tungsten@host1:/home/tungsten/tungsten-replicator-7.1.2-81
# Options for the alpha data service
tools/tpm configure alpha \
--enable-thl-ssl=true \
--install-directory=/opt/continuent \
--java-keystore-password=password \
--java-truststore-password=password \
--master=host1 \
--members=host1,host2,host3 \
--replication-password=password \
--replication-user=tungsten \
--start=true \
--topology=master-slave
```

The output includes all of the tpm configure commands necessary to rebuild the configuration. It includes all default, dataservice and host specific configuration settings. Review this output and make changes as needed until you are satisfied.

## 9.3.5. Installation

After you have prepared the configuration file, it is time to install.

```
shell> ./tools/tpm install
```

This will install all services defined in configuration. The installation will be done as explained in Section 9.2, "Processing Installs and Upgrades". This will include the full set of `--members` [413], `--slaves` [423], and `--master` [412].

### 9.3.5.1. Installing a set of specific services

```
shell> ./tools/tpm install alpha,bravo
```

All hosts included in the alpha and bravo services will be installed. The installation will be done as explained in Section 9.2, "Processing Installs and Upgrades".

### 9.3.5.2. Installing a set of specific hosts

```
shell> ./tools/tpm install --hosts=host1,host2
```

Only `host1` and `host2` will be installed. The installation will be done as explained in Section 9.2, "Processing Installs and Upgrades".

## 9.3.6. Upgrades from a Staging Directory

This process must be run from the staging directory in order to run properly. Determine where the current software was installed from.

```
shell> tpm query staging
tungsten@staging-host:/opt/continuent/software/tungsten-replicator-7.1.2-81
```

This outputs the hostname and directory where the software was installed from. Make your way to that host and the parent directory before proceeding. Unpack the new software into the `/opt/continuent/software` directory and make it your current directory.

```
shell> tar zxf tungsten-replicator-7.1.2-81.tar.gz
shell> cd tungsten-replicator-7.1.2-81
```

#### Warning

> Before performing and upgrade, please ensure that you have checked the Appendix B, *Prerequisites*, as software and system requirements may have changed between versions and releases.

Before any update, the current configuration must be known. If the `$CONTINUENT_PROFILES` or `$REPLICATOR_PROFILES` environment variables were used in the original deployment, these can be set to the directory location where the configuration was stored.

Alternatively, the update can be performed by fetching the existing configuration from the deployed directory by using the tpm fetch command:

```
shell> ./tools/tpm fetch --reset --directory=/opt/continuent \
    --hosts=host1,autodetect
```

This will load the configuration into the local staging directory. Review the current configuration before making any configuration changes or deploying the new software.

```
shell> ./tools/tpm reverse
```

This will output the current configuration of all services defined in the staging directory. You can then make changes using tpm configure before pushing out the upgrade. Run tpm reverse again before tpm update to confirm your changes were loaded correctly.

```
shell> ./tools/tpm configure service_name ...
shell> ./tools/tpm update --replace-release
```

#### Important

> The use of `--replace-release` is not mandatory for minor configuration changes. however it is highly recommended when upgrading between versions.
>
> Using this option will ensure that underlying metadata and property files are cleanly rebuilt, thus ensuring any new or deprecated properties between releases are correctly added/removed acordingly.

This will update the configuration file and then push the updates to all hosts. No additional arguments are needed for the tpm update command since the configuration has already been loaded.

## 9.3.7. Configuration Changes from a Staging Directory

Where, and how, you make configuration changes depends on where you want the changes to be applied.

Making Configuration Changes to the Current Host

You may make changes to a specific host from the `/opt/continuent/tungsten` directory.

```
shell> ./tools/tpm update service_name --thl-log-retention=14d
```

This will update the local configuration with the new settings and restart the replicator. You can use the tpm help update command to see which components will be restarted.

```
shell> ./tools/tpm help update | grep thl-log-retention
--thl-log-retention How long do you want to keep THL files?
```

If you make changes in this way then you must be sure to run tpm fetch from your staging directory prior to any further changes. Skipping this step may result in you pushing an old configuration from the staging directory.

Making Configuration Changes to all hosts

This process must be run from the staging directory in order to run properly. Determine where the current software was installed from.

```
shell> tpm query staging
tungsten@staging-host:/opt/continuent/software/tungsten-replicator-7.1.2-81
```

This outputs the hostname and directory where the software was installed from. Make your way to that host and directory before proceeding.

```
shell> ./tools/tpm fetch --reset --directory=/opt/continuent \
    --hosts=host1,autodetect
```

This will load the configuration into the local staging directory. Review the current configuration before making any configuration changes or deploying the new software.

```
shell> ./tools/tpm reverse
```

This will output the current configuration of all services defined in the staging directory. You can then make changes using tpm configure before pushing out the upgrade. Run tpm reverse again before tpm update to confirm your changes were loaded correctly.

```
shell> ./tools/tpm configure service_name ...
shell> ./tools/tpm update
```

This will update the configuration file and then push the updates to all hosts. No additional arguments are needed for the tpm update command since the configuration has already been loaded.

## 9.3.8. Converting from INI to Staging

If you currently use the INI installation method and wish to convert to using the Staging method, there is currently no easy way to do that. The procedure involves uninstalling fully on each node, then reinstalling from scratch.

If you still wish to convert from the INI installation method to using the Staging method, use the following procedure:

1. On the staging node, extract the software into `/opt/continuent/software/{extracted_dir}`

   ```
   shell> cd /opt/continuent/software
   shell> tar zxf tungsten-replicator-7.1.2-81.tar.gz
   ```

2. Create the text file `config.sh` based on the output from tpm reverse:

   ```
   shell> cd tungsten-replicator-7.1.2-81
   shell> tpm reverse > config.sh
   ```

   Review the new `config.sh` script to confirm everything is correct, making any needed edits. When ready, create the new configuration:

   ```
   shell> sh config.sh
   ```

   Review the new configuration:

   ```
   shell> tools/tpm reverse
   ```

   See Section 9.3, “tpm Staging Configuration” for more information.

3. On all nodes, uninstall the Tungsten software:

   **Warning**

   Executing this step WILL cause an interruption of service.

```
shell> tpm uninstall --i-am-sure
```

4. On all nodes, rename the `tungsten.ini` file:

```
shell> mv /etc/tungsten/tungsten.ini /etc/tungsten/tungsten.ini.old
```

5. On the staging node only, change to the extracted directory and execute the tpm install command:

```
shell> cd /opt/continuent/software/tungsten-replicator-7.1.2-81
shell> ./tools/tpm install
```

# 9.4. tpm INI File Configuration

tpm can use an INI file to manage host configuration. This is a fundamental difference from the normal model for using tpm. When using an INI configuration, the tpm command will only work with the local server.

In order to configure Tungsten on your server using an INI file you must still complete all of the Appendix B, *Prerequisites*. Copying SSH keys between your servers is optional but setting them up makes sure that certain scripts packaged with Continuent Tungsten will still work.

## 9.4.1. Creating an INI file

When using an INI configuration, installation and updates will still be done using the tpm command. Instead of providing configuration information on the command line, the tpm command will look for an INI file in three files:

1. `$ENV{HOME}/tungsten.ini`.
2. `/etc/tungsten/tungsten.ini`
3. `/etc/tungsten.ini`

tpm will automatically search all `tungsten*.ini` files within the `/etc/tungsten` directory.

An alternative directory can be searched using `--ini` [368] option to tpm. This option can also be used to specify a specific ini file if you choose to name the file something different, for example `--ini /my/directory/myconfig.ini`

The INI file(s) must be readable by the tungsten system user.

Here is an example of a `tungsten.ini` file that would setup a simple dataservice.

```
shell> vi /etc/tungsten/tungsten.ini
```

```
[alpha]
master=host1
members=host1,host2,host3
connectors=host1,host2,host3

[defaults]
application-user=app_user
application-password=secret
application-port=3306
replication-user=tungsten
replication-password=secret
replication-port=13306
start-and-report=true
user=tungsten
```

The property names in the INI file are the same as what is used on the command line. Simply remove the leading `--` characters and add it to the proper section. Each section in the INI file replaces a single tpm configure call. The section name inside of the square brackets is used as the service name. In the case of the `[defaults]` section, this will act like the tpm configure defaults command.

Include any host-specific options in the appropriate section. This configuration will only apply to the local server, so there is no need to put host-specific settings in a different section.

## 9.4.2. Installation with INI File

Once you have created the `tungsten.ini` file, the tpm command will recognize it and use it for configuration. Unpack the software into `/opt/continuent/software` and run the tpm install command.

```
shell> cd /opt/continuent/software/tungsten-replicator-7.1.2-81
shell> ./tools/tpm install
```

or

```
shell> cd /opt/continuent/software/tungsten-replicator-7.1.2-81
shell> ./tools/tpm install --ini /my/directory/myconfig.ini
```

The tpm command will read the `tungsten.ini` file and setup all dataservices on the current server.

## 9.4.3. Upgrades with an INI File

Use the tpm update command to upgrade to the latest version.

```
shell> cd /opt/continuent/software
shell> tar zxf tungsten-replicator-7.1.2-81.tar.gz
shell> cd tungsten-replicator-7.1.2-81
shell> ./tools/tpm update --replace-release
```

### Important

The use of `--replace-release` is not mandatory for minor configuration changes. however it is highly recommended when upgrading between versions.

Using this option will ensure that underlying metadata and property files are cleanly rebuilt, thus ensuring any new or deprecated properties between releases are correctly added/removed acordingly.

After unpacking the new software into the staging directory, the tpm update command will read the `tungsten.ini` configuration and install the new software. All services will be stopped and the new services will be started.

During the lifetime of the cluster, switches may happen and the current Primary may well be a different node than what is reflected in the static ini file in the `master=` line. Normally, this difference is ignored during an update or an upgrade.

However, if a customer has some kind of procedure (i.e. automation) which hand-edits the ini configuration file `master=` line at some point, and such hand-edits do not reflect the current reality at the time of the update/upgrade, an update/upgrade will fail and the cluster may be left in an indeterminate state.

### Warning

The best practice is to NOT change the `master=` line in the INI configuration file after installation.

There is still a window of opportunity for failure. The update will continue, passing the `CurrentTopologyCheck` test and potentially leaving the cluster in an indeterminate state if the `master=` option is set to a hostname that is not the current Primary or the current host.

## 9.4.4. Configuration Changes with an INI file

The tpm update also allows you to apply any configuration changes. Start by making any necessary changes to the `tungsten.ini` file. Then proceed to running tpm update.

```
shell> cd /opt/continuent/tungsten
shell> ./tools/tpm update
```

This will read the `tungsten.ini` file and apply the settings. The tpm command will identify what services likely need to be restarted and will just restart those. You can manually restart the desired services if you are unsure if the new configuration has been applied.

## 9.4.5. Converting from Staging to INI

If you currently use the Staging installation method and wish to convert to using INI files, use the following procedure.

You can also try using the script in Section 9.4.6, “Using the `translatetoini.pl` Script”.

1. Create the text file `/etc/tungsten/tungsten.ini` on each node. They will normally all be the same.

   ```
   shell> sudo mkdir /etc/tungsten
   shell> sudo chown -R tungsten: /etc/tungsten
   shell> chmod 700 /etc/tungsten
   shell> touch /etc/tungsten/tungsten.ini
   shell> chmod 600 /etc/tungsten/tungsten.ini
   ```

   Each section in the INI file replaces a single tpm configure call. The section name inside of [square brackets] is used as the service name. In the case of the [defaults] section, this will act like the tpm configure defaults command. The property names in the INI file are the same as what is used on the command line. Simply remove the leading -- characters and add it to the proper section.

For example, to seed the `tungsten.ini` file, use the output of tpm reverse:

```
shell> tpm reverse > /etc/tungsten/tungsten.ini
```

Edit the new ini file and clean it up as per the rules above. For example, using vim:

```
shell> vim /etc/tungsten/tungsten.ini
:%s/tools\/^tpm configure /[/g
:%s/^--//g
:%s/\s*\\$//g
```

> **Important**
>
> In the above example, you MUST manually add the trailing square bracket ] to the end of the defaults tag and to the end of every service name section. Just search for the opening square bracket [ and make sure there is a matching closing square bracket for every one.

See Section 9.4.1, "Creating an INI file" for more information.

2. On every node, extract the software into `/opt/continuent/software/{extracted_dir}`

> **Warning**
>
> Make sure you have the same release that is currently installed.

```
shell> cd /opt/continuent/software
shell> tar zxf tungsten-replicator-7.1.2-81.tar.gz
```

3. On each node, change to the extracted directory and execute the tpm command:

```
shell> cd /opt/continuent/software/tungsten-replicator-7.1.2-81
shell> ./tools/tpm update
```

This will read the `tungsten.ini` file and apply the settings. The tpm command will identify what services likely need to be restarted and will just restart those. You can manually restart the desired services if you are unsure if the new configuration has been applied.

## 9.4.6. Using the `translatetoini.pl` Script

You can download a script from the documentation library, translatetoini.pl. You must have a copy of Perl installed to be able to execute the script.

To use the script, you can either run the script and paste in the staging output, or pipe the output from tpm reverse directly into the script. When supplying the staging output, you should supply the output from the within the configured staging directory. For example:

```
shell> ./tools/tpm reverse|../translatetoini.pl
```

The script will create the file `tungsten.ini` in the current directory containing the converted output.

To change the destination, use the `--filename` option:

```
shell> ./tools/tpm reverse|../translatetoini.pl --filename=t.ini
```

You can also combine multiple staging configurations into a single INI conversion by appending to an existing INI file by adding the `--append` option:

```
shell> ./tools/tpm reverse|../translatetoini.pl --append
```

You should always check the INI file before using it for a live installation to ensure that all of the options and parameters have been identified and configured properly.

A training video is available on how to perform the staging to INI file conversion using the translatetoini.pl script:

Click here for a video of the INI conversion procedure, showing the full process from start to finish...

# 9.5. tpm Commands

All calls to tpm will follow a similar structure, made up of the command, which defines the type of operation, and one or more options.

```
shell> tpm command [sub command] [tpm options] [command options]
```

The command options will vary for each command. The core tpm options are:

Table 9.2. tpm Core Options

| Option | Description |
|---|---|
| --force [335], -f [335] | Do not display confirmation prompts or stop the configure process for errors |
| --help [335], -h [335] | Displays help message |
| --info [335], -i [335] | Display info, notice, warning and error messages |
| --notice [335], -n [335] | Display notice, warning and error messages |
| --preview [335], -p [335] | Displays the help message and preview the effect of the command line options |
| --profile file [335] | Sets name of config file |
| --quiet [335], -q [335] | Only display warning and error messages |
| --verbose [335], -v [335] | Display debug, info, notice, warning and error messages |

– --force [335]

Forces the deployment process to complete even if there are warning or error messages that would normally cause the process the to fail. Forcing the installation also ignores all confirmation prompts during installation and always attempts to complete the process.

– --help [335]

Displays the help message for tpm showing the current options, commands and version information.

– --info [335]

Changes the reporting level to include information, notice, warning and error messages. Information level messages include annotations of the current process and stage in the deployment, such as configuration or generating files and configurations. This shows slightly more information than the default, but less than the full debug level offered by --verbose [335].

– --notice [335]

Sets the output level to include notice, warning, and error messages. Notice level messages include information about further steps or actions that should be taken, or things that should be noted without indicating a failure or error with the configuration options select.

– --preview [335]

– --profile file [335]

Specify the name of the configuration file to be used. This can be useful if you are performing multiple configurations or deployments from the same staging directory. The entire configuration and deployment information is stored in the file before installation is started. By specifying a different file you can have multiple deployments and configurations without requiring separate staging directories.

– --quiet [335]

Changes the error reporting level so that only warning and error messages are displayed. This mode can be useful in automated deployments as it provides output only when a warning or error exists. All other messages, including informational ones, are suppressed.

– --verbose [335]

Displays a much more detailed output of the status and progress of the deployment. In verbose mode, tpm annotates the entire process describing both what it is doing and all debug, warning and other messages in the output.

The tpm utility handles operations across all hosts in the dataservice. This is true for simple and composite dataservices. The coordination requires SSH connections between the hosts according to the Appendix B, *Prerequisites*. There are two exceptions for this:

1. When the --hosts [409] argument is provided to a command; that command will only be carried out on the hosts listed. Multiple hosts may be given as a comma-separated list. The names used in the --members [413], --slaves [423], --master [412] arguments should be used when calling --hosts [409].

2. When you are using an INI configuration file (see Section 9.4, "tpm INI File Configuration") all calls to tpm will only affect the current host.

The installation process starts in a staging directory. This is different from the installation directory where Tungsten Cluster will ultimately be placed but may be a sub-directory. In most cases we will install to /opt/continuent but use /opt/continuent/software as a staging directory. The release package should be unpacked in the staging directory before proceeding. See the Section B.2, "Staging Host Configuration" for instructions on selecting a staging directory.

Table 9.3. tpm Commands

| Option | Description |
|---|---|
| ask | Ask tpm to provide values from the common configuration |
| cert | |
| configure | Configure a data service within the global configuration |
| delete-service | Delete a replication service or a composite datasource |
| diag | Obtain diagnostic information |
| fetch | Fetch configuration information from a running service |
| firewall | Display firewall information for the configured services |
| help | Show command help information |
| install | Install a data service based on the existing and runtime parameters |
| mysql | Open a connection to the configured MySQL server |
| post-process | Reset the THL for a host |
| purge-thl | Reset the THL for a host |
| query | Query the active configuration for information |
| report | Generate a security report for all available communication channels on a per-node basis |
| reset | Reset the cluster on each host |
| reset-thl | Reset the THL for a host |
| uninstall | Uninstall software from host(s) |
| update | Update an existing configuration or software version |
| validate | Validate the current configuration |
| validate-update | Validate the current configuration and update |

## 9.5.1. tpm ask Command

tpm ask can be used to query values from the common configuration

Usage:

```
tpm ask [args] [context] {value_or_function_name} [argument]
```

`[context]` : The optional context may be one of: `common` (default), `keys`, `all` or `function`

If you specify a context of:

- `common` or no context: Extract the variable from the perl common object
- `certs` : Display information about certificates including expiry dates
- `certtpm, certlocations` : Display reference information about the `security.properties` file
- `all`: Extract every available variable from the perl common object
- `keys`: Extract all available variable names from the perl common object
- `functions`: Extract the available function names from the perl common object
- `function`: Extract the return value of the named function from within the perl_common object. You may specify an optional argument to the function.

Examples:

```
shell> tpm ask keys
shell> tpm ask all
shell> tpm ask functions
shell> tpm ask services
shell> tpm ask isCAA
```

```
shell> tpm ask function managerEnabled
shell> tpm ask function cleanBool false
```

Arguments:

Table 9.4. tpm ask Common Options

| Option | Description |
|---|---|
| `--allstages` | Display stages of the replictors for all roles |
| `--api` | Use the v2 API REST interface instead of the command line when possible |
| `--debug`, `-d` | Debug Mode. |
| `--dsrole` | Display local datasource role |
| `--dsstate` | Display local datasource state |
| `--help`, `-h` | |
| `--info`, `-i` | |
| `--nodeinfo` | Display state and role of local datasource and replicator |
| `-p {path}`, `--path {path}` | Pass full path to replicator executables e.g. /opt/replicator/tungsten/tungsten-replicator/bin |
| `--quiet`, `-q` | |
| `--stages` | Display stages of the local replictor |
| `--test`, `-t` | |
| `--thl {file}` | Path and name of thl executable (Ignores --path if also supplied) |
| `--trepctl {file}` | Path and name of trepctl executable (Ignores --path if also supplied) |
| `--trrole` | Display local replicator role |
| `--trstate` | Display local replicator state |
| `-v` | Verbose output. |

## 9.5.2. tpm cert Command

### Introduction

The tpm cert command is designed to streamline the generation of Tungsten-specific security files for use by the tpm install and tpm update commands.

The tpm cert command minimizes the complexity of handling certificates but cannot remove it entirely. For the best results, contact Continuent Support and get help from the experts before using this command.

By default, the tpm install command will take care of all security files in new installations (upgrades preserve existing) when `disable-security-controls=false` or when the option is ommitted when installing v7.0.0 or later.

You may need to provide your own certificates, or handle enabling security at a later time and install with `disable-security-controls=true`. If so, tpm cert is for you.

- Cert SOURCE files are in the "certsdir" `$CONTINUENT_ROOT/generated/`
- Cert RUNNING files are in the "security directory" `$CONTINUENT_ROOT/share/`

By default, all commands use source files located in "certsdir". To read running files in the "security directory", use `--running` (or `-r`).

To use custom values/paths, create a file called `tungsten.env` in the "security directory" and populate the variables as needed.

#### Important

There is no way to rotate security files with zero downtime. This is a database server limitation because once the first database server process is restarted with the new certs, the cluster will not be able to communicate with it. The same goes for the tpm update step if done before the database server process restarts. Once all database servers are using the new certs and all cluster nodes have been updated, everything will be able to communicate properly and the operation will be done.

### Important

LIMITATION: At this time, tpm cert cannot be run if the Tungsten software is not yet installed. This will be fixed in the next release.

## 9.5.2.1. tpm cert Usage

USAGE: `tpm cert {action} {typeSpec} [args]`

Table 9.5. tpm cert Read-Only Actions

| Option | Description |
|---|---|
| `aliases`, `al` | Display alias names from one or more files. |
| `ask`, `as` | Display various information. |
| `cat`, `ca` | Display key files. |
| `diff`, `d` | Compare running files with generated files. |
| `example`, `ex` | Display example files. |
| `info`, `in` | Display metadata about a security file as JSON. |
| `list`, `li` | Show the contents of a security file. |
| `ls` | List a directory. |
| `help`, `h` | Display short help text. |

Table 9.6. tpm cert Write Actions

| Option | Description |
|---|---|
| `import`, `ad`, `add`, `im` | Add one or more typeSpecs into another. |
| `backup`, `ba` | Backup one or more key directories and files. |
| `cp`, `ch`, `changepass` | Change the storepass for one or more files. |
| `clean`, `cl` | Delete all files in a directory. |
| `gen`, `cr`, `create`, `g` | Generate various security files. |
| `vi`, `v` | Edit the file. |
| `rm`, `rem`, `remove` | Delete a specific alias from a security file. |
| `swap`, `ro`, `rotate`, `sw` | Replace an existing entry with one from another file. |

Table 9.7. tpm cert Arguments

| Option | Description |
|---|---|
| `--count`, `-c` | Display an integer count of aliases found instead of the actual aliases. |
| `--debug`, `-d` | Displays debug-level status messages. |
| `--dir` | Specify the target directory to store files in. |
| `--dryrun`, `-n` | Do not execute the command, display what would be done instead. |
| `--extra`, `-x` | Display the command to be run before executing, and other additional information when available. |
| `--help`, `-h` | Displays a help message. |
| `--i-am-sure` | Confirm you want the DESTRUCTIVE operation (delete/rotate) to proceed without an interactive pause. |
| `--info`, `-i` | Displays info-level status messages. |
| `--livetls` | Use the running tungsten_tls_keystore.jks in $CONTINUENT_ROOT/share/. You may not use --tls and --livetls together. |
| `--long`, `-l` | Display verbose output in keytool and openssl and other areas. |
| `--mysqldir` | Specify the target directory to store MySQL-specific files in. |
| `--quiet`, `-q` | Hides status output whenever possible. |

| Option | Description |
|---|---|
| `--running`, `-r` | Use the running files from $CONTINUENT_ROOT/share/ instead of the certs source directory $CONTINUENT_ROOT/generated/. |
| `--tls` | Specify a source TLS typeSpec (either tls_keystore or TLS_FILE). |
| `-v` | Displays verbose-level status messages. |

To see more detailed help on each action, you can use the following commands:

```
shell> tpm cert h {action}
```

### 9.5.2.2. tpm cert {typeSpec}, Defined

- A typeSpec is a case-sensitive, unique string that identifies a key security file, possibly located in different subdirectories. Examples include:
  - keystore = tungsten_keystore.jks
  - connector_keystore = tungsten_connector_keystore.jks
- `{typeSpec}` can be either a single string or a comma-separated list with no spaces, for example:

```
shell> tpm cert info connector_truststore
shell> tpm cert aliases jgroups_keystore,tls_keystore
```

- Use tpm cert help typespec to see the standard typeSpec
- Use tpm cert help {Action} to see the typeSpec for that action
- Different `{Action}`s have different typeSpecs
- Some typeSpecs are groups of other typeSpecs
- There are three classes of typeSpec:
  1. Pre-Defined Source Tungsten-specific files
     - These files are located in `$CONTINUENT_ROOT/generated/` and are populated by tpm cert gen
  2. Pre-Defined "Running" Tungsten & MySQL-specific files
     - Tungsten-specific files are located in `$CONTINUENT_ROOT/share/`, and are populated by tpm [install|update]
     - MySQL-specific files are located in the MySQL datadir by default, and can be populated with tpm cert gen mysqlcerts
     - The Tungsten running files are accessed by adding `--running` (or `-r`) on the command line while using the same typeSpec - for example,
       - tpm cert info ct displays `$CONTINUENT_ROOT/generated/connector_truststore.ts`
       - tpm cert info ct -r displays `$CONTINUENT_ROOT/share/connector_truststore.ts`
  3. User-Defined Source and Target Files
     - These files are typically located in `$BASE_DIR/` and are configured via `$CONTINUENT_ROOT/share/tungsten.env`. They are also populated by tpm cert gen
     - Example `BASE_DIR` values, possibilities are endless, as long as the Tungsten OS user (usually 'tungsten') has write access to that directory or the ability to create that directory:
       - /etc/tungsten/secure
       - /var/lib/mysql
       - /home/tungsten/certs

### 9.5.2.3. `{typeSpec}` definitions

PRE-Defined RUNNING Tungsten-specific files

- Access these via cli arg `--running` (or `-r`)

- Located in `$CONTINUENT_ROOT/share/`, populated by tpm [install|update]

| Option | Description |
|---|---|
| `keystore`,`jk`,`ke` | tungsten_keystore.jks |
| `truststore`,`ts`,`tr` | tungsten_truststore.ts |
| `connector_keystore`,`cj`,`ck` | tungsten_connector_truststore.ts |
| `connector_truststore`,`ct` | tungsten_connector_truststore.ts |
| `thl_keystore`,`tk` | tungsten_thl_keystore.jks |
| `thl_truststore`,`tt` | tungsten_thl_truststore.ts |
| `tls_keystore`,`tl` | tungsten_tls_keystore.jks |
| `jgroups_keystore`,`jg` | tungsten_jgroups_keystore.jceks |
| `tls_passwordstore`,`pw` | passwords.store |
| `jmx_passwordstore`,`jm` | jmxremote.access |

PRE-Defined RUNNING Tungsten-specific MySQL files

- Located in MySQL datadir by default, populated by tpm cert gen mysqlcerts

| Option | Description |
|---|---|
| `mysqlca`,`mc` | ca.pem |
| `mysqlcert`,`mt` | client-cert.pem |
| `mysqlkey`,`mk` | client-key.pem |

PRE-Defined SOURCE Tungsten-specific files

- Located in `$CONTINUENT_ROOT/generated/`, populated by tpm cert gen]

| Option | Description |
|---|---|
| `keystore`,`jk`,`ke` | tungsten_keystore.jks |
| `truststore`,`ts`,`tr` | tungsten_truststore.ts |
| `connector_keystore`,`cj`,`ck` | tungsten_connector_truststore.ts |
| `connector_truststore`,`ct` | tungsten_connector_truststore.ts |
| `thl_keystore`,`tk` | tungsten_thl_keystore.jks |
| `thl_truststore`,`tt` | tungsten_thl_truststore.ts |
| `tls_keystore`,`tl` | tungsten_tls_keystore.jks |
| `jgroups_keystore`,`jg` | tungsten_jgroups_keystore.jceks |
| `tls_passwordstore`,`pw` | passwords.store |
| `jmx_passwordstore`,`jm` | jmxremote.access |
| `mysqlp12`,`p1` | client-cert.p12 |

USER-Defined Source and Target Files

- Typically located in `$BASE_DIR/` and configured via `$CONTINUENT_ROOT/share/tungsten.env`

| Option | Description |
|---|---|
| `CRT_FILE`,`CR` | .crt file generated from the .pfx file |
| `CERT_PEM_FILE`,`CE` | .pem file generated from the .crt file |
| `CA_PEM_FILE`,`CA` | .pem CA file generated by MySQL |
| `KEY_FILE`,`KE` | .key and .key.encrypted generated from the .pfx file |
| `P12_FILE`,`P1` | .p12 file generated from the .pem and .key files |
| `JKS_FILE`,`JK` | Tungsten Keystore, i.e. tungsten_keystore.jks |
| `TS_FILE`,`TS` | Tungsten Truststore, i.e. tungsten_truststore.ts |

| Option | Description |
|---|---|
| CJKS_FILE,CJ | Tungsten Connector Keystore, i.e. tungsten_connector_keystore.jks |
| CTS_FILE,CT | Tungsten Connector Truststore, i.e. tungsten_connector_truststore.ts |
| TJKS_FILE,TJ | Tungsten THL Keystore, i.e. tungsten_thl_keystore.jks |
| TTS_FILE,TT | Tungsten THL Truststore, i.e. tungsten_thl_truststore.ts |
| JGROUPS_FILE,JG | Tungsten jGroups Keystore, i.e. tungsten_jgroups_keystore.jceks |
| TLS_FILE,TL | Tungsten TLS Keystore, i.e. tungsten_tls_keystore.jks |
| PW_FILE,PW | Tungsten TLS Password Store, i.e. passwords.store |
| JMX_FILE,JM | Tungsten RMI/JMX Password Store, i.e. jmxremote.access |
| PFX_FILE,PF | Source .pfx file, i.e. mysql.pfx |

The following "Convenience tags" can be used in place of specific {typeSpec} options.

Table 9.8. Convenience tags

| Option | Description |
|---|---|
| all, a | Varies based on the following factors: If tungsten.env exists, then use the User-defined files from it, else use the pre-defined files from $CONTINUENT_ROOT/generated, or from $CONTINUENT_ROOT/share if --running is specified on the cli. |
| batch, b | runs typeSpec defined in BATCH envvar, comma-separated |
| pfx | Runs PFX_FILE,user |
| tungsten, tu | Runs pre-defined: tl,jg,jk,ts,cj,ct,tj,tt,pw,jm |
| user | Runs user-defined TL,JG,JK,TS,CJ,CT,TJ,TT,PW,JM |

### 9.5.2.4. {passwordSpec} definitions

Used to specify passwords in a standard format for the openssl and keytool commands so that you do not need to remember the command-specific syntax, which is different for each command.

Password Specifications MUST be one of:

- env:VARNAME - has to be pre-defined and exported as an EnvVar in the $CONTINUENT_ROOT/share/tungsten.env file.
- run:typeSpec - available as part of the running Tungsten config for {typeSpec} of:
  - keystore|k
  - truststore|t
  - connector_keystore|ck
  - connector_truststore|ct
- pass:yourPasswordHere - specify the actual password string and be sure to escape any special characters from the shell.

If the PasswordSpec provided does not begin with env:, run: or pass: it will be rejected.

### 9.5.2.5. tpm cert: Getting Started - Basic Examples

- Display each typeSpec and the list of aliases found in it:

```
shell> tpm cert aliases all
shell> tpm cert aliases all -c
shell> tpm cert aliases all -l -x
```

- Display the file information as JSON, including SHA and Expiration date:

```
shell> tpm cert info tungsten
shell> tpm cert info tungsten -r
```

- Use keytool or openssl to display the contents of the file:

```
shell> tpm cert list truststore
shell> tpm cert list truststore -l -x
```

- Display an example of needed INI entries for custom work:

```
shell> tpm cert example ini
shell> tpm cert ex i
```

- Edit the /etc/tungsten/tungsten.ini file using vi:

```
shell> tpm cert vi ini
shell> tpm cert v i
```

- Display an example of needed tungten.env entries for custom work:

```
shell> tpm cert example env
shell> tpm cert ex env 1
shell> tpm cert ex e 2
```

- Generate a new $CONTINUENT_ROOT/share/tungten.env file:

```
shell> tpm cert gen env 1
```

- Edit the $CONTINUENT_ROOT/share/tungsten.env file using vi:

```
shell> tpm cert vi env
shell> tpm cert v e
```

## 9.5.2.6. tpm cert: Getting Started - Functional Database Cert Rotation Example

The philosophy here is that the cert rotation work is done on a single cluster node. We will call this the "work node".

The secret to getting certs to work with a cluster is to make sure that you copy all of the MySQL database certs and Tungsten security files from the work node to the rest of the nodes at the correct point in the process. If all the cluster and database cert files are not the same across all nodes, the cluster will fail.

In the following functional example, we demonstrate the actual steps to rotate the database certs in a standalone 5-node Tungsten cluster.

When doing a database cert rotation, multiple files must be regenerated:

- Five (5) required MySQL security files (may be more created than needed):
  1. ca.pem
  2. client-cert.pem
  3. client-key.pem
  4. server-cert.pem
  5. server-key.pem
- One (1) .p12 file to represent the database client cert:
  1. client-cert.p12
- Four (4) Tungsten-specific files:
  1. tungsten_keystore.jks
  2. tungsten_truststore.ts
  3. tungsten_truststore.ts
  4. tungsten_connector_truststore.ts

Next, confirm that both MySQL and Tungsten are configured to use the proper files:

- `/etc/my.cnf` entries:

```
[mysqld]
ssl-ca=/etc/mysql/certs/ca.pem
ssl-cert=/etc/mysql/certs/server-cert.pem
```

```
ssl-key=/etc/mysql/certs/server-key.pem
require_secure_transport=ON
```

> **Important**
>
> This example assumes that the database certs are located in `/etc/mysql/certs` on all cluster database nodes.

- `/etc/tungsten/tungsten.ini` entries:

```
datasource-mysql-ssl-ca=/etc/mysql/certs/ca.pem
datasource-mysql-ssl-cert=/etc/mysql/certs/client-cert.pem
datasource-mysql-ssl-key=/etc/mysql/certs/client-key.pem
```

> **Important**
>
> This example assumes that no `/opt/continuent/share/tungsten.env` file exists.

*** FUNCTIONAL PROCEDURE EXAMPLE STARTS HERE ***

## ** On a single node (db1) as the tungsten OS user **

- Backup the old database certs from `/etc/mysql/certs/` to `/opt/continuent/backups/`:

```
tpm cert backup mysql
```

- Clean out any old database certs so new certs are generated:

```
sudo rm /etc/mysql/certs/*.pem
```

- Verify all old database certs are gone:

```
ls -l /etc/mysql/certs/*.pem
```

- Generate new mysql certs:

```
tpm cert gen mysqlcerts -x
```

- Set proper ownership and permissions for Tungsten access:

```
sudo chown -R mysql: /etc/mysql/certs/
sudo chmod -R g+r /etc/mysql/certs/
```

- Verify new database certs:

```
ls -l /etc/mysql/certs/*.pem
```

- Copy new database certs to all other database nodes:

```
for i in 2 3 4 5; do sudo rsync -avc --delete /etc/mysql/certs/ db$i:/etc/mysql/certs/; done
```

- Backup any previously-generated cluster certs to `/opt/continuent/backups/`:

```
tpm cert backup gen
```

- Backup the running cluster certs to `/opt/continuent/backups/`:

```
tpm cert backup share
```

- Regenerate the MySQL .p12 file and the needed Tungsten security files, using --livetls to specify the running TLS cert file (`/opt/continuent/share/tungsten_tls_keystore.jks`):

```
tpm cert gen p12,ke,ts,ck,ct --livetls
```

- Examine the new files:

```
tpm cert info p12,ke,ts,ck,ct
```

- Copy new files to ALL other cluster nodes:

```
tpm copy --gen
    ~OR~
for i in 2 3 4 5; do rsync -avc --delete /opt/continuent/generated/ db$i:/opt/continuent/generated/; done
```

- Set the cluster policy to MAINENANCE

```
tpm policy -m
```

## ** On all cluster nodes as the tungsten OS user **

- Stop the cluster processes:

```
stopall
```

- Restart the database server process:

```
sudo systemctl restart mysqld
    ~OR~
sudo service mysqld restart
```

- Identify the Tungsten software staging directory:

```
cd `tpm query staging| cut -d: -f2`
```

- Update the Tungsten software to use the new certs, which will restart all Tungsten processes:

```
tools/tpm update -i --replace-release
```

- When all updates have completed, start the cluster software:

```
startall
```

- When all nodes have been updated and started, wait 30 seconds, then test:

```
echo ls | cctrl
tpm connector
```

## ** On a single node as the tungsten OS user **

- When all of the above tests are ok, set the cluster policy to AUTOMATIC:

```
tpm policy -a
```

- Test again:

```
echo ls | cctrl
tpm connector
```

*** FUNCTIONAL PROCEDURE EXAMPLE ENDS HERE ***

## 9.5.2.7. tpm cert: Getting Started - Conversion to Custom-Generated Security Files Example

In the following example, we take an existing cluster that was installed using Tungsten-self-generated security files and convert it to use custom-generated security files.

The basic security file conversion steps are:

- Ensure three tpm config options exist and point to the correct files:

```
shell> tpm query config | grep datasource_mysql_ssl
    datasource_mysql_ssl_ca (tpm option: datasource-mysql-ssl-ca)
    datasource_mysql_ssl_cert (tpm option: datasource-mysql-ssl-cert)
    datasource_mysql_ssl_key (tpm option: datasource-mysql-ssl-key)
```

- Generate all standard security files and place into {certsdir} on ONE node only

```
shell> tpm cert gen all
```

- Display new files info as json for standard cert files in {certsdir}

```
shell> tpm cert info all
```

- Copy new files to all other cluster nodes

```
shell> tpm copy --gen
```

- Display example tungsten.ini contents

```
shell> tpm cert example ini
```

- Add those lines to the `/etc/tungsten/tungsten.ini` on ALL cluster nodes

```
shell> tpm cert vi ini
```

- Place the cluster into MAINTENANCE mode

```
shell> tpm policy -m
```

- Display the directory that the software was installed from:

```
shell> tpm query staging
```

- Update the cluster software to use the new security files in {certsdir} which will restart the Tungsten processes:

```
shell> cd {staging_dir_from_above}
shell> tools/tpm update --replace-release
```

- Once all cluster updates are done, return the cluster to AUTOMATIC mode

```
shell> tpm policy -a
```

## 9.5.2.8. tpm cert: Getting Started - Advanced Example

You may want to provide your own certificates, or have installed with `disable-security-controls=true`, and now wish to enable security. If so, tpm cert is for you.

In the following advanced example, we will rotate the database certs using a source .pfx file.

### --- Summary ---

- Populate the `tungsten.env` file
- Generate the security files defined in `tungsten.env`
- Add new options to the `tungsten.ini` to match
- Update the software using the new security settings

### --- Details ---

- Displays example `tungsten.env` contents

```
tpm cert example env
```

- Create a new `$CONTINUENT_ROOT/share/tungsten.env` file, which defaults to example id 1:

```
tpm cert gen env 2
```

- Run vi `$CONTINUENT_ROOT/share/tungsten.env`

```
tpm cert vi env
    export BASE_DIR=/etc/tungsten/secure
    export BATCH="pfx2p12,JK,TS,CJ,CT"
```

- Display variables set in `$CONTINUENT_ROOT/share/tungsten.env`

```
tpm cert ask env
```

- Displays example `tungsten.ini` contents

```
tpm cert example ini
```

- Run vi `/etc/tungsten/tungsten.ini`

```
tpm cert vi ini
    java-keystore-path=/etc/tungsten/secure/tungsten_keystore.jks
    java-truststore-path=/etc/tungsten/secure/tungsten_truststore.ts
    java-connector-keystore-path=/etc/tungsten/secure/tungsten_connector_keystore.jks
    java-connector-truststore-path=/etc/tungsten/secure/tungsten_connector_truststore.ts
```

- Generate all cert files in the BATCH envvar defined in the `tungsten.env` file:

```
tpm cert gen batch --livetls -x
```

- Display info as json all cert files in the BATCH envvar defined in the `tungsten.env` file:

```
tpm cert info P12,JK,TS,CJ,CT
```

- Display the extracted package staging directory that the software was installed from:

```
tpm query staging
```

- Update the software to use the new cert files in {certsdir}:

```
cd {staging_dir}
tools/tpm update --replace-release
```

### 9.5.2.9. Using tpm cert add

The `add` action is used to add one or more typeSpec into another.

Usage: `tpm cert add|ad|import|im {sourceTypeSpec} {targetTypeSpec} [alias] [passwordSpec]`

- `sourceTypeSpec` may be either a single argument or a comma-delimited list (no spaces)
- `alias` is optional and will be assigned the defaults of:
  - `tls` for any source TLS files
  - `mysql` for any source MySQL cert files
- `passwordSpec` is optional and defaults to `env:STORE_PASS`

Examples:

```
shell> tpm cert add mysqlp12 keystore mysql
shell> tpm cert add P12_FILE connector_keystore mysql
shell> tpm cert add tls JKS_FILE -r
shell> tcert add P12,tls JK -r -x
```

`{sourceTypeSpec}` and `{targetTypeSpec}` may not both be batch.

`{sourceTypeSpec}` for add must be one of (specName|shortcut):

- `CA_DIR|DI`
- `CA_PEM_FILE|CA`
- `CERT_PEM_FILE|CE`
- `mysqlca|mc`
- `mysqlp12|my`
- `P12_FILE|P1`
- `TLS_FILE|TL`
- `tls_keystore|tl`
- `batch|b`

`{targetTypeSpec}` for add must be one of (specName|shortcut):

- `JKS_FILE|JK`
- `TS_FILE|TS`
- `CJKS_FILE|CJ`
- `CTS_FILE|CT`
- `TJKS_FILE|TJ`
- `TTS_FILE|TT`

- `keystore|jk|ke`
- `truststore|ts|tr`
- `connector_keystore|cj|ck`
- `connector_truststore|ct`
- `thl_keystore|tj|tk`
- `thl_truststore|tt`
- `batch|b`

## 9.5.2.10. Using tpm cert aliases

The `aliases` action is a Read-Only action and can be used to display the aliases for a given typeSpec, additionally, arguments can be supplied for further detail, as explained below:

Argument Usage with `aliases`

- Use `--count|-c` for a numeric quantity of aliases found only.
- Use `--extra|-x` to also show the file fullpath
- Use `--long|-l` to display one alias per line, cannot be used with `--count|-c`

Examples:

```
shell> tpm cert aliases keystore
shell> tpm cert aliases keystore,truststore
shell> tpm cert aliases ke,tr
shell> tpm cert aliases CRT_FILE
shell> tpm cert aliases CR
shell> tpm cert aliases running
shell> tpm cert aliases all
```

## 9.5.2.11. Using tpm cert ask

The tpm cert ask can be used to show specific information according to the typeSpec provided.

Examples:

```
shell> tpm cert ask certs
shell> tpm cert ask c
shell> tpm cert ask locations
shell> tpm cert ask l
shell> tpm cert ask tpm
shell> tpm cert ask t
shell> tpm cert ask all
```

### Table 9.9. typeSpecs for tpm cert ask

| Option | Description |
|---|---|
| `all`, `a` | |
| `certs`, `c` | |
| `env`, `e` | |
| `locations`, `l` | |
| `tpm`, `t` | |

## 9.5.2.12. Using tpm cert backup

The `backup` action backups up one or more files/directories represented by the typeSpec.

- All backups are created with an ISO timestamp extension (.YYYYMMDDHHMMSS)
- Use `--extra|-x` to show the command executed

Usage: `tpm cert backup {typeSpec}`

`{typeSpec}` for backup must be one of:

- `base|b`

  DIR /opt/continuent/generated/

  TO Parent dir if base dir is not the same as the certs dir, otherwise /opt/continuent/backups/

- `certs|c`

  DIR /opt/continuent/generated/

  TO /opt/continuent/backups/

- `env|e`

  FILE /opt/continuent/share/tungsten.env

  TO /opt/continuent/share/

- `ini|i`

  FILE /etc/tungsten/tungsten.ini

  TO /etc/tungsten/

- `mysql|m`

  DIR /

  TO /opt/continuent/backups/

- `share|s`

  DIR /opt/continuent/share/

  TO /opt/continuent/backups/

Examples

```
shell> tpm cert backup base
shell> tpm cert backup certs
shell> tpm cert backup env
shell> tpm cert backup ini
shell> tpm cert backup my
shell> tpm cert backup share
```

## 9.5.2.13. Using tpm cert cat

The `cat` action displays the contens of the specified file, requires that `{typeSpec}` be provided

Usage: `tpm cert cat {typeSpec}`

`{typeSpec}` must be one of:

- `ini|i`
- `env|e`
- `all|a`

Examples:

```
shell> tpm cert cat all
shell> tpm cert cat env
shell> tpm cert cat ini
```

## 9.5.2.14. Using tpm cert changepass

tpm cert changepass allows you to change the password for a given `{typeSpec}`

Usage: `tpm cert changepass {typeSpec} {oldPasswordSpec} {newPasswordSpec}`

Examples:

```
shell> tpm cert changepass JK pass:tungsten env:STORE_PASS
```

In addition to the standard `{typeSpec}` (Execute tpm cert help typespec for a full list) the following `{typeSpec}`s are also available:

- `batch|b` : Runs `typeSpec` defined in BATCH envvar, comma-separated

Password Specifications MUST be one of:

- `env:VARNAME` - has to be pre-defined and exported as an `EnvVar` in the `$CONTINUENT_ROOT/share/tungsten.env` file.
- `run:typeSpec` - available as part of the running Tungsten config for `{typeSpec}` of:
  - `keystore|k`
  - `truststore|t`
  - `connector_keystore|ck`
  - `connector_truststore|ct`
- `pass:yourPasswordHere` - specify the actual password string and be sure to escape any special characters from the shell.

If the `PasswordSpec` provided does not begin with `env:`, `run:` or `pass:` it will be rejected.

## 9.5.2.15. Using tpm cert clean

The `clean` removes all files from the directory or directories represented by the `{typeSpec}` provided.

Use `--extra|-x` to show the command executed

Usage: `tpm cert clean {typeSpec}`

`{typeSpec}` must be one of:

- `base|b` - Clean out the directory defined as `$BASE_DIR`
- `certs|c|gen|g` - Clean out the `$CONTINUENT_ROOT/generated` directory

Examples:

```
shell> tpm cert clean base
shell> tpm cert clean certs
```

## 9.5.2.16. Using tpm cert diff

The `diff` can be used to compare the generated versus running files for `{typeSpec}`

Usage: `tpm cert diff {typeSpecLeft} {typeSpecRight}`

- Add `-n` to just see the files that will be compared
- Add `-r` (or `--running`) to use the file in the "security directory" `$CONTINUENT_ROOT/share/` for the `{typeSpec}` instead of from "certsdir" `$CONTINUENT_ROOT/generated/`

In addition to the standard `{typeSpec}` (Execute tpm cert help typespec for a full list) the following `{typeSpec}`s are also available:

- `batch|b` (runs typeSpec defined in BATCH envvar, comma-separated)

Examples:

```
shell> tpm cert clean base
shell> tpm cert diff keystore -n
shell> tpm cert diff keystore
shell> tpm cert diff JK jk -n
shell> tpm cert diff JK jk
shell> tpm cert diff JK jk -r -n
shell> tpm cert diff JK jk -r
```

### 9.5.2.17. Using tpm cert example

tpm cert example can be used to display examples for specific `typeSpecs`

Usage: `tpm cert example {typeSpec} [topicID]`

Examples:

```
shell> tpm cert example ini
shell> tpm cert ex env
shell> tpm cert ex env 1
```

Table 9.10. typeSpecs for tpm cert example

| Option | Description |
|---|---|
| `all`, `a` | Show all sample entries from both ini and env |
| `env`, `e` | Show tungsten.env example entries, Supply optional topicID of 1 or 2 to limit display to Example 1 or Example 2 |
| `ini`, `i` | Show tungsten.ini example entries |

### 9.5.2.18. Using tpm cert info

tpm cert info shows the information as json for the given `{typeSpec}`

Usage: `tpm cert info {typeSpec}`

Examples:

```
shell> tpm cert info keystore
shell> tpm cert info keystore,truststore
shell> tpm cert info ke,tr
shell> tpm cert info CRT_FILE
shell> tpm cert info CR
shell> tpm cert info running
shell> tpm cert info all
```

For a full list of the standard `{typeSpec}` execute tpm cert help typespec

### 9.5.2.19. Using tpm cert list

tpm cert info displays the file(s) represented by the `{typeSpec}`

Usage: `tpm cert list {typeSpec}`

Additionally, you can use the following arguments:

- `--long|-l` to display the file verbosely
- `--extra|-x` to show the command executed

Examples:

```
shell> tpm cert list keystore
shell> tpm cert list keystore,truststore
shell> tpm cert list ke,tr
shell> tpm cert list ke,tr --long
shell> tpm cert list ke,tr -x
shell> tpm cert list running
shell> tpm cert list all
```

For a full list of the standard `{typeSpec}` execute tpm cert help typespec

### 9.5.2.20. Using tpm cert gen

tpm cert gen is used to generate the specified typeSpec file(s). This is the core action since the tpm cert command is designed to streamline the generation of Tungsten-specific security files for use by the tpm install and tpm update commands.

Basic examples:

```
shell> tpm cert gen all
```

```
shell> tpm cert gen batch
shell> tpm cert gen mysqlcerts
shell> tpm cert gen mysqlp12
shell> tpm cert gen tungsten
shell> tpm cert gen user
```

Advanced examples:

```
shell> tpm cert gen P12_FILE,JK,TS,CJ,CT
shell> tpm cert gen pfx2p12,JK,TS,CJ,CT
shell> tpm cert gen pfx2p1
shell> tpm cert gen pfx2key
shell> tpm cert gen pfx2crt
shell> tpm cert gen crt2pem
shell> tpm cert gen P12_FILE
```

In addition to the standard `{typeSpec}` (Execute tpm cert help typespec for a full list) the following `{typeSpec}`s are also available:

### Table 9.11. typeSpecs for tpm cert gen

| Option | Description |
|---|---|
| `all`, `a` | Runs P12_FILE,tungsten |
| `batch`, `b` | Runs typeSpec defined in BATCH envvar, comma-separated |
| `crt2pem` | Requires database cert file CRT_FILE {.crt}. Generates .pem from .crt |
| `env`, `e` | Generates $CONTINUENT_ROOT/share/tungsten.env |
| `mysqlcerts` | Runs 'sudo mysql_ssl_rsa_setup'. See note below. |
| `mysqlp12` | Generates a p12 file from the configured MySQL client cert files if they exist [client-cert.pem, client-key.pem and ca.pem]. The new file will be created in {certsdir}: $CONTINUENT_ROOT/generated/client-cert.p12 |
| `pfx` | Runs pfx2p12,tungsten |
| `pfx2crt` | Requires database cert file PFX_FILE {.pfx}, CERT_PASS. Generates .crt from .pfx |
| `pfx2key` | Requires database cert file PFX_FILE {.pfx}, CERT_PASS. Generates .key and .key.encrypted files from .pfx file |
| `pfx2p12` | Requires database cert file PFX_FILE {.pfx}, STORE_PASS, CERT_PASS optional. Runs pfx2key,pfx2crt,crt2pem,P12_FILE |
| `tungsten`, `tu` | Runs pre-defined: tl,jg,jk,ts,cj,ct,tj,tt,pw,jm |
| `user`, `u` | Runs user-defined: TL,JG,JK,TS,CJ,CT,TJ,TT,PW,JM |

#### Note

`CERT_PASS` is optional for Tungsten because usually database client certs do not have a password See Section 6.10.2, “Configure Tungsten<>Database Secure Communication”

#### Note

Further detail on `mysqlcerts` typeSpec:

mysqlcerts runs sudo mysql_ssl_rsa_setup, please see https://dev.mysql.com/doc/refman/5.7/en/mysql-ssl-rsa-setup.html

From the above docs: "If openssl is present, mysql_ssl_rsa_setup looks for default SSL and RSA files [ca.pem,server-cert.pem, server-key.pem] in the MySQL data directory specified by the --datadir option, or the compiled-in data directory if the --datadir option is not given. If any of those files are present, mysql_ssl_rsa_setup creates no SSL files. Otherwise, it invokes openssl to create them, plus some additional files:

- ca.pem : Self-signed CA certificate
- ca-key.pem : CA private key
- server-cert.pem : Server certificate
- server-key.pem : Server private key

- client-cert.pem : Client certificate
- client-key.pem : Client private key

### 9.5.2.21. Using tpm cert remove

tpm cert remove deletes an existing certificate from a keystore or trustore

Usage: `tpm cert remove {typeSpec} {certAlias}`

Examples:

```
shell> tpm cert remove
```

In addition to the standard `{typeSpec}` (Execute tpm cert help typespec for a full list) the following `{typeSpec}`s are also available:

- `batch|b` (runs typeSpec defined in BATCH envvar, comma-separated)

### 9.5.2.22. Using tpm cert rotate

The `rotate` action is used to replace an existing entry with one from another file.

This has the same effect as executing tpm cert add -f

Usage: `tpm cert rotate|ro|swap|sw {sourceTypeSpec} {targetTypeSpec} [alias] [passwordSpec]`

For the list of available `{typeSpec}` for this action, see Section 9.5.2.9, "Using tpm cert add"

Examples:

```
shell> tpm cert rotate mysqlp12 keystore mysql
shell> tpm cert rotate P12_FILE connector_keystore mysql
shell> tpm cert ro tls thl_keystore
shell> tpm cert ro CA_DIR connector_keystore,connector_truststore
shell> tpm cert ro CA_DIR CJ,CT -x
```

### 9.5.2.23. Using tpm cert vi

tpm cert vi can be used to edit the file specified by the `typeSpec`

Usage: `tpm cert vi {typeSpec}`

Examples:

```
shell> tpm cert vi ini
shell> tpm cert v e
```

Table 9.12. typeSpecs for tpm cert vi

| Option | Description |
|---|---|
| `env`, `e` | Edit tungsten.env |
| `ini`, `i` | Edit tungsten.ini |

## 9.5.3. tpm configure Command

The configure command to tpm creates a configuration file within the current profiles directory

## 9.5.4. tpm delete-service Command

This command was introduced in version 6.1.13.

The tpm delete-service command allows you to cleanly remove a dataservice from your cluster, or a single replication service from a stand-alone replicator installation.

The tpm delete-service command will know if it is being run from the Clustering software of the Replicator software, and will act accordingly

See the table below for a list of valid arguments:

Table 9.13. tpm delete-service Common Options

| Option | Description |
|---|---|
| `--api` | Use the v2 API REST interface instead of the command line when possible |
| `--auto`, `-A` | Automatically execute any needed commands that it is possible to handle. |
| `--debug`, `-d` | Debug Mode. |
| `--i-have-run-tpm-update` | For Staging-method installations, pass this flag to confirm that the `tools/tpm update`command has already been run from the staging directory |
| `--help`, `-h` | |
| `--i-am-sure` | Bypass the 'Are You Sure?' prompt when using --auto. |
| `--info`, `-i` | |
| `-n {file}`, `--newini {file}` | Pass the fullpath (and filename) to an INI file to be used at the 'Edit INI' step |
| `-f` | Pass the force flag to tpm. |
| `-p {path}`, `--path {path}` | Pass full path to replicator executables e.g. /opt/replicator/tungsten/tungsten-replicator/bin |
| `--quiet`, `-q` | |
| `--test`, `-t` | |
| `--thl {file}` | Path and name of thl executable (Ignores --path if also supplied) |
| `--trepctl {file}` | Path and name of trepctl executable (Ignores --path if also supplied) |
| `-v` | Verbose output. |

A number of options determine the behavior of this command and these are outlined below.

Usage for tpm delete-service

```
shell> tpm delete-service [args] {service_name} [configuration_service_name]
```

`{service_name}` is the Replicator service name as seen in trepctl services.

`[configuration_service_name]` is the Replicator service name as seen in tpm reverse, and only needed for standalone Replicator installations where the service name in the configuration is not the same as the actual replication service name, i.e. in cluster-extractor topologies where there is a cluster-alias employed.

The default behavior is to display the needed commands for the admin to execute manually.

To use this tool in a fully automated manner:

- specify `--auto`
- include `--i-am-sure` to bypass most interactive prompts
- include `--newini {filename}` otherwise you will be prompted to edit the INI file in the vi editor before proceeding

Use-cases

- Tungsten Replicator: i.e. for Standalone, Cluster-Extractor, Fan-In and Multi-Site/Active-Active topologies where there is a discrete Replicator running outside of a Cluster (needs Replicator workflow)

Workflows

1. Replicator Service

```
shell> trepctl -all-services offline
shell> rm tungsten-replicator/conf/static-SERVICE.properties
shell> rm tungsten-replicator/conf/.static-SERVICE.properties.orig
shell> replicator restart
shell> trepctl -all-services online

# Remove the various directories after the restart so that the replicator no longer has the dirs open :
```

```
shell> rm -rf thl/SERVICE/
shell> rm -rf relay/SERVICE/
shell> trepctl -all-services online
```

## 9.5.5. tpm diag Command

The tpm diag command will create a TGZ file including log files, current dataservice status and a number of OS metrics.

```
shell> tpm diag
NOTE  >> host1 >> Diagnostic information written to /home/tungsten/tungsten-diag-2013-10-09-21-04-23.tgz
```

The operation of tpm diag differs between installation types (Staging vs INI). This is outlined below:

- With Staging-method deployments, the tpm diag command can be issued in two ways:
  - The tpm diag command alone will obtain diagnostics from all hosts in the cluster.
  - The tpm diag --hosts host1,host2,hostN command will obtain diagnostics from the specified host(s) only.
- Within an INI installation, the behaviour will depend on a number of factors, these are outlined below
  - For versions prior to 5.3.7, and version 6.0.0 to 6.0.4
    - The tpm diag command alone will attempt to obtain diagnostics from all hosts in the cluster if ssh has been configured and the other hosts can be reached.
  - For versions 5.3.7 to 5.4.0, and versions 6.0.5 onwards
    - The tpm diag command alone will ONLY obtain diagnostics from the local host on which the command is executed.
    - The tpm diag --hosts host1,host2,hostN command will obtain diagnostics from the specified host(s) only.
    - The tpm diag -a|--allhosts command will attempt to obtain diagnostics from all hosts in the cluster if ssh has been configured and the other hosts can be reached. The output of tpm diag will provide feedback detailing the hosts that were reached.

The structure of the created file will depend on the configured hosts, but will include all the logs for each accessible host configured in individual directories for each host.

Additional options

It is possible to limit the amount of information gathered by tpm diag by optionally skipping individual gather subroutines, or skipping entire groups. These are outlined below

- `--list` : Print all diagnostic gathering groups and associated subroutines for use with `--skip` and `--skipgroups`
- `--include` : Specify a comma-separated list of subroutines to include (NO spaces). Anything not listed will be skipped.
- `--skip` : Specify a comma-separated list of subroutines to skip (NO spaces)
- `--groups` : Specify a comma-separated list of subroutine groups to include (NO spaces). Anything not listed will be skipped.
- `--skipgroups` : Specify a comma-separated list of subroutine groups to skip (NO spaces)
- `--skipsudo|--nosudo` : Prevent operations using the sudo command. Using this option may result in some diagnostic collection failing.

Examples

```
shell> tpm diag --list

GROUP: SUBROUTINE(s)
cluster:    cctrlClusterValidate cctrlHistoryFile cctrlLong cctrlPing cctrlStatus
general:    confDirs logFiles miscFiles
mysql:      etcMysql etcMycnf etcMyInclude etcMyIncludedirs getMysqlCommands mysqlErrorLog
os:         getOSCommands etcHosts cpuInfo etcSystemRelease
replicator: thlInfo thlIndex trepctlPerf trepctlQuick trepctlStatus trepctlStatusJSON
tpm:        etcTungsten tpmDiff tpmReverse tpmValidate
```

```
shell> tpm diag --skip=getOSCommands
```

```
shell> tpm diag --skipgroups=os
```

tungsten_send_diag

If the host you are running the diag from has external internet connectivity, you may also wish to consider using tungsten_send_diag. This will run tpm diag for you and automatically upload the resulting file to Continuent Support. For more information on using this, see Section 8.33, "The tungsten_send_diag Script"

## 9.5.6. tpm fetch Command

There are some cases where you would like to review the configuration or make changes prior to the upgrade. In these cases it is possible to fetch the configuration and process the upgrade as different steps.

```
shell> ./tools/tpm fetch \
    --directory=/opt/continuent \
    --hosts=host1,autodetect
```

This will load the configuration into the local staging directory. You can then make changes using tpm configure before pushing out the upgrade.

The tpm fetch command supports the following arguments:

- `--hosts` [409]

  A comma-separated list of the known hosts in the cluster. If `autodetect` is included, then tpm will attempt to determine other hosts in the cluster by checking the configuration files for host values.

- `--user` [429]

  The username to be used when logging in to other hosts.

- `--directory` [404]

  The installation directory of the current Tungsten Cluster installation. If `autodetect` is specified, then tpm will look for the installation directory by checking any running Tungsten Cluster processes.

## 9.5.7. tpm firewall Command

The tpm firewall command displays port information required to configured a firewall. When used, the information shown is for the current host:

```
shell> tpm firewall
To host1
---------------------------------------------------------------------------
From application servers
From connector servers        13306
From database servers         2112, 13306
```

The information shows which ports, on which hosts, should be opened to enable communication.

## 9.5.8. tpm help Command

The tpm help command outputs the help information for tpm showing the list of supported commands and options.

```
shell> tpm help
Usage: tpm help [commands,config-file,template-file] [general-options] [command-options]
-------------------------------------------------------------------------------------
General options:
-f, --force                   Do not display confirmation prompts or stop the configure »
                              process for errors
-h, --help                    Displays help message
--profile file                Sets name of config file (default: tungsten.cfg)
-p, --preview                 Displays the help message and preview the effect of the »
                              command line options
-q, --quiet                   Only display warning and error messages
-n, --notice                  Display notice, warning and error messages
-i, --info                    Display info, notice, warning and error messages
-v, --verbose                 Display debug, info, notice, warning and error messages
...
```

To get a list of available configuration options, use the config-file subcommand:

```
shell> tpm help config-file
#######################################################################
# Config File Options
#######################################################################
config_target_basename        [tungsten-replicator-7.1.2-81_pid10926]
```

```
deployment_command            Current command being run
remote_package_path           Path on the server to use for running tpm commands
deploy_current_package        Deploy the current Tungsten package
deploy_package_uri            URL for the Tungsten package to deploy
deployment_host               Host alias for the host to be deployed here
staging_host                  Host being used to install
...
```

## 9.5.9. tpm install Command

The tpm install command performs an installation based on the current configuration (if one has been previously created), or using the configuration information provided on the command-line.

For example:

```
shell> ./tools/tpm install alpha\
    --topology=master-slave \
    --master=host1 \
    --replication-user=tungsten \
    --replication-password=password \
    --home-directory=/opt/continuent \
    --members=host1,host2,host3 \
    --start
```

Installs a service using the command-line configuration.

```
shell> ./tools/tpm configure alpha\
    --topology=master-slave \
    --master=host1 \
    --replication-user=tungsten \
    --replication-password=password \
    --home-directory=/opt/continuent \
    --members=host1,host2,host3
shell> ./tools/tpm install alpha
```

Configures the service first, then performs the installation steps.

During installation, tpm checks for any host configuration problems and issues, copies the Tungsten Cluster software to each machine, creates the necessary configuration files, and if requests, starts and reports the status of the service.

If any of these steps fail, changes are backed out and installation is stopped.

## 9.5.10. tpm keep Command

tpm keep command is designed to streamline saving the current Tungsten Replicator position for each service in a variety of formats:

- dsctl get as `.json`
- dsctl get -ascmd as `.cmd`
- mysqldump as `.dmp`

Usage:

```
tpm keep [args]
```

### Table 9.14. tpm keep Options

| Option | Description |
|---|---|
| `--all`, `-a` | Dump all available tungsten_* schemas instead of using the list from trepctl services |
| `--debug`, `-d` | Displays debug-level status messages. |
| `--dir` | Specify the target directory to store files in. |
| `--dryrun`, `-n` [335] | Do not execute the command, display what would be done instead. |
| `--nodump` | Skip the mysqldump step. |
| `--extra`, `-x` | Display the command to be run before executing. |
| `--help` [335], `-h` [335] | Displays a help message. |
| `--info` [335], `-i` [335] | Displays info-level status messages |

| Option | Description |
| --- | --- |
| `--list`, `-l` | Display tracking schema instead of saving to disk. |
| `--long`, `-L` | Display other additional information when available. |
| `--noget` | Skip the dsctl get step. |
| `--quiet` [335], `-q` [335] | Hides status output whenever possible |
| `--service`, `-s` | Specify the service name(s) to save as comma separated list, default: all |
| `--verbose` [335], `-v` [335] | Displays verbose-level status messages. |

There are two distinct gathering steps:

Step 1: Run dsctl twice per service (.json and .cmd)

- Skip this step with `--noget`
- Step 1 requires that the database server be running

Step 2: Run mysqldump once per schema (.dmp)

> **Note**
>
> This step will be skipped for non-MySQL Replicator only targets

- Skip this step with `--nodump`
- Step 2 requires that the database server be running
- Gather all available tungsten_* schemas using `--all`

## 9.5.11. tpm mysql Command

This will open a MySQL CLI connection to the local MySQL server using the current values for `--replication-user` [421], `--replication-password` [420] and `--replication-port` [421].

```
shell> tpm mysql
```

This command will fail if the mysql utility is not available or if the local server does not have a running database server.

## 9.5.12. tpm post-process Command

The tpm post-process Command assists with the graceful maintenance of the static cross-site replicator configuration files on disk.

The tpm post-process command performs the following steps:

- Locates, reads and parses the various INI configuration files.

  Below is the list of possible INI files and file match patterns:

  - `{$HOME}/tungsten.ini`
  - `/etc/tungsten/tungsten*.ini`

    > **Note**
    >
    > Please note that all files in the `/etc/tungsten` directory starting with `tungsten` and ending in `.ini` will be used.

  - `/etc/tungsten.ini`
- Identifies cross-site services that are local to the node, as well as the main local service.

  > **Warning**
  >
  > Only Replicator options and properties for specific entry types are currently supported.

  The supported stanzas are as follows:

- `{service}.replicator`
- `{service}_from_{service}`

Below is an example INI file showing section identifiers only:

```
[defaults]
[defaults.replicator]
[east]
[east.replicator]
[west]
[east_from_west]
[west_from_east]
```

Of the above example section identifiers, only the following would be used by tungsten_post_process:

- `east.replicator`
- `east_from_west`
- `west_from_east`

The remaining example section identifiers (`defaults`, `defaults.replicator`, `east` and `west`) would be ignored.

- Gathers configuration options and properties defined for each identified local service.
- Locates the associated configuration file on disk and gets the existing value for the option key.
- Interactively prompts for confirmation. This may be bypassed using the `-y` option to tpm post-process.
- Filter out any files that are not static replicator configurations.
- The tpm post-process command will then do one edit-in-place per ini option per local service.
- The script will check to make sure the value has been updated.
- Finally, there will be a summary message and helpful instructions about next steps. This may be bypassed using the `-q` option to tpm post-process.

Table 9.15. tpm post-process Options

| Option | Description |
|---|---|
| `--debug`, `-d` | Enable additional debug output. |
| `--quiet`, `-q` | Prevent final message from being output. |

Below is a sample session:

```
shell> tpm post-process
About to update the following config file(s):

/opt/continuent/tungsten/tungsten-replicator/conf/static-east_from_north.properties
/opt/continuent/tungsten/tungsten-replicator/conf/static-east_from_west.properties

Do you wish to continue? [y/N] y

Done!

tungsten_post_process updated 0 configuration options successfully and skipped 6

Pick one node and set Maintenance mode so the manager does not try to bring the replicator online when you don't want it to:

  echo "set policy maintenance" | cctrl

Then take the Replicator offline gracefully on one node:

  trepctl -all-services offline

Restart the Replicator:

  replicator restart

Bring all services online:

  trepctl -all-services online
```

```
Check the status:

  trepctl services

Repeat as needed.

When all nodes are done, pick one node and execute:

  echo "set policy automatic" | cctrl
```

### Note

The `post-process` will be automatically called by tpm during `update` and `install` to ensure any cross-site specific configuration is applied at the correct time

## 9.5.13. tpm purge-thl Command

The tpm option `purge-thl` is designed to assist with identifying the safe removal of THL based on the following rules:

- Gather the last applied seqno from all Replica nodes and take the lowest one
- Find the current THL file which contains that seqno, then locate the previous one
- construct a thl purge command to remove thl thru the last seqno in the prev file

A replicator service name name may be optionally specified as the last argument, for example:

```
shell> tpm purge-thl -A alpha
```

The default behavior is to display the needed commands for the admin to execute manually.

The tungsten_purge_thl can be used as a read-only option to generate the commands required to purge manually.

Table 9.16. tpm purge-thl Options

| Option | Description |
|---|---|
| `--auto`, `-A` | Automatically execute any needed commands that it is possible to handle. Edits to the configuration are not possible at this time. |
| `--debug`, `-d` | Enable additional debug output. |
| `--help`, `-h` | Show help text |
| `--i-am-sure` | Bypass the "Are You Sure?" prompt when using --auto. |
| `--info`, `-i` | |
| `--path`, `-p` | Use to supply full path to replicator executables used for thl and trepctl if not in default location. |
| `--quiet`, `-q` | |
| `--test`, `-t` | |
| `--thl` | Use to supply full path to replicator executables used for thl if not in default location. Ignores --path if supplied |
| `-d`, `--debug` | Use to supply full path to replicator executables used for trepctl if not in default location. Ignores --path if supplied |
| `--verbose`, `-v` | |

## 9.5.14. tpm query Command

The query command provides information about the current tpm installation. There are a number of subcommands to query specific information:

- tpm query config — return the full configuration values
- tpm query dataservices — return the list of dataservices
- tpm query default — return the list of configured default values
- tpm query deployments — return the configuration of all deployed hosts

- tpm query manifest — get the manifest information
- tpm query modified-files — return the list of files modified since installation by tpm
- tpm query staging — return the staging directory from where Tungsten Cluster was installed
- tpm query values — return the list of configured values
- tpm query version — get the version of the current installation

### 9.5.14.1. tpm query config

Returns a list of all of the configuration values, both user-specified and implied within the current configuration. The information is returned in the form a JSON value:

```
shell> tpm query config
{
  "__system_defaults_will_be_overwritten__": {
...
  "staging_directory": "/home/tungsten/tungsten-replicator-7.1.2-81",
  "staging_host": "tr-ms1",
  "staging_user": "tungsten"
}
```

### 9.5.14.2. tpm query dataservices

Returns the list of configured dataservices that have, or will be, installed:

```
shell> tpm query dataservices
alpha                        : PHYSICAL
```

### 9.5.14.3. tpm query deployments

Returns a list of all the individual deployment hosts and configuration information, returned in the form of a JSON object for each installation host:

```
shell> tpm query deployments
{
  "config_target_basename": "tungsten-replicator-7.1.2-81_pid22729",
  "dataservice_host_options": {
    "alpha": {
      "start": "true"
    }
...
  "staging_directory": "/home/tungsten/tungsten-replicator-7.1.2-81",
  "staging_host": "tr-ms1",
  "staging_user": "tungsten"
}
```

### 9.5.14.4. tpm query manifest

Returns the manifest information for the identified release of Tungsten Replicator, including the build, source and component versions, returned in the form of a JSON value:

```
shell> tpm query manifest
{
  "date": "Wed Jun  3 16:54:45 UTC 2020",
  "fullVersion": "6.1.4",
  "git": {
    "URL": "file:///volumes/data/bamboo/home/xml-data/build-dir/_git-repositories-cache/dfe910bfa6ded0f410e78a8215885afe1a539172",
    "branch": "cnt-6.1.4-merge",
    "revision": "8b7939809ae685cf459d76d052a3c9916112306b"
  },
  "host": "bamboo",
  "hudson": {
    "URL": "",
    "buildId": 44,
    "buildNumber": 44,
    "buildTag": "",
    "jobName": ""
  },
  "product": "Tungsten Replicator",
  "productCode": "tungsten.replicator",
  "release": "tungsten-replicator-6.1.4-44",
  "releaseBaseName": "tungsten-replicator",
  "userAccount": "bamboo",
```

```
  "version": {
    "major": 6,
    "minor": 1,
    "revision": 4
  }
}
```

### 9.5.14.5. tpm query modified-files

Shows the list of configuration files that have been modified since the installation was completed. Modified configuration files cannot be overwritten during an upgrade process, using this command enables you identify which files contain changes so that these modifications can be manually migrated to the new installation. To restore or replace files with their original installation, copy the `.filename.orig` file.

### 9.5.14.6. tpm query staging

Returns the host and directory from which the current installation was created:

```
shell> tpm query staging
tungsten@host1:/home/tungsten/tungsten-replicator-7.1.2-81
```

This can be useful when the installation host and directory from which the original configuration was made need to be updated or modified.

### 9.5.14.7. tpm query version

Returns the version for the identified version of Tungsten Cluster:

```
shell>  tpm query version
7.1.2-81
```

## 9.5.15. tpm report Command

The purpose of tpm report is to provide easy access to all of the settings that pertain to a specific topic.

Usage:

```
tpm report [args]
```

The default (and only) topic is the security stance. More topics will be added over time.

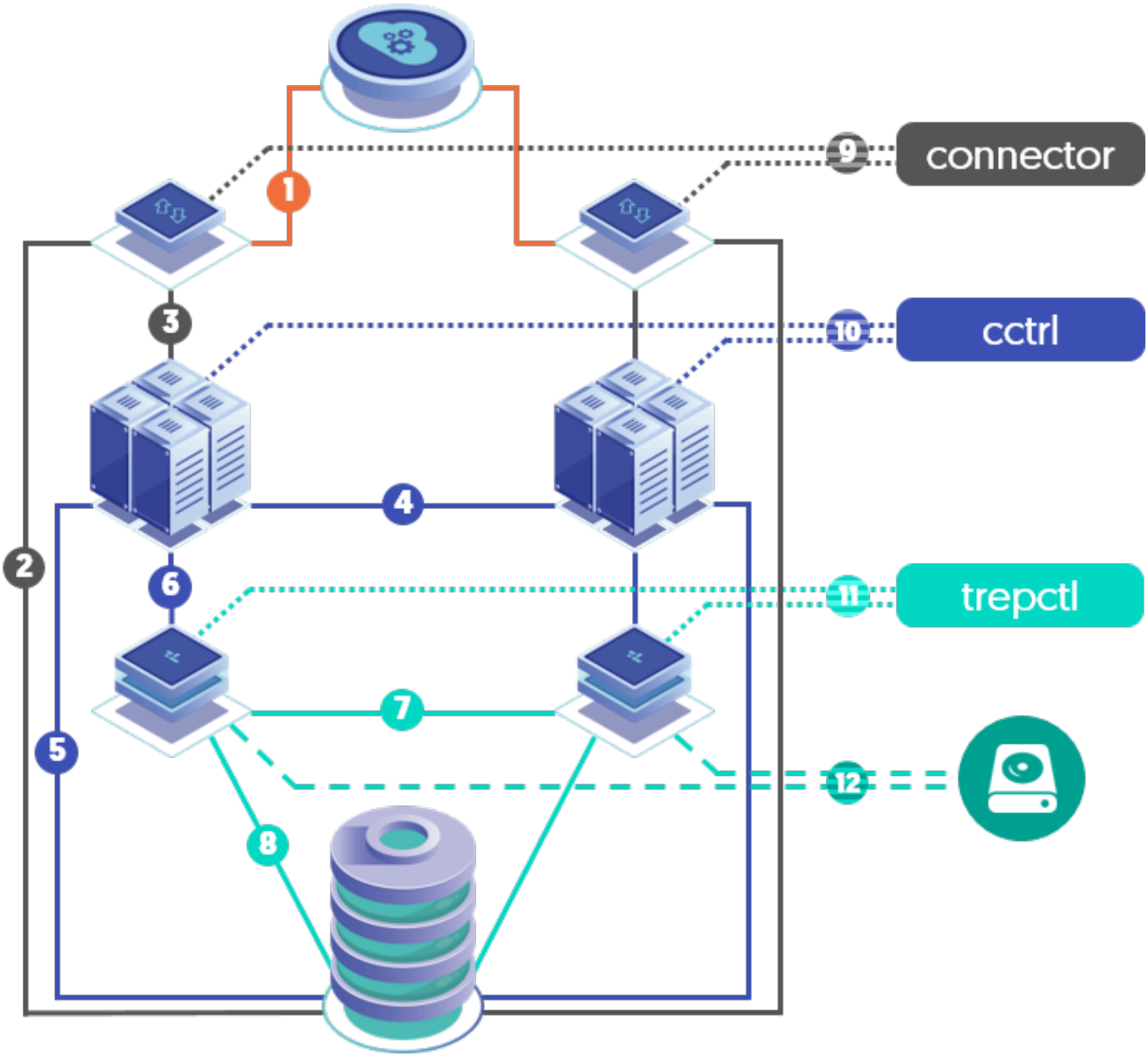
Each topic contains a set of numbered reports. View the list of reports for any topic using `--list` (or `-l`). For example:

```
shell> tpm report --list
>>> Security Reports <<<

1. Application to Connector (mysql)
Communications from the client application to the Connector port

2. Connector to Database (mysql)
Communications from the Connector to the Database port

3. Connector to Manager (proprietary (routerGateway))
Communications from the Connector to the Manager

4. Manager to Manager (rmi/jmx, jgroups)
Communications from Manager to Manager

5. Manager to Database (mysql)
Communications from Manager to Database

6. Manager to Replicator (rmi/jmx)
Communications from Manager to Replicator

7. Replicator to Replicator (thl)
Communications from Replicator to Replicator

8. Replicator to Database (mysql)
Communications from Replicator to Database

9. connector Command to the Connector (rmi/jmx)
Communications from the connector cli command to the to local Connector process

10. cctrl Command to the Manager (rmi/jmx)
Communications from the cctrl cli command to the to Manager process

11. trepctl Command to the Replicator (rmi/jmx)
Communications from the trepctl cli command to the to Replicator process
```

```
12. Replicator On Disk THL (proprietary (thl encryption))
THL files on disk are encrypted

13. SSL-Specific `tpm` Options for the API (http,https)
Options used by tpm to enable or disable the API for each component
```

The following graphic provides a visual representation to the various communication channels within the cluster and how they are mapped to the various "levels" presented above.

Figure 9.3. Internals: Cluster Communication Channels




To display just a specific report, specify it using the `--report`. For example:

```
shell> tpm report --report 1

TUNGSTEN SECURITY REPORT as of Thu May  5 20:37:17 UTC 2022

----------------------------------------------------------------------
>> Comm Channel 1:  Application to Connector <<
----------------------------------------------------------------------

=> CHANNEL INFORMATION:
```

```
Connector Bridge Mode: OFF
Application-to-Connector SSL is Enabled

=> TPM OPTIONS:
connector-ssl-capable = true
enable-connector-client-ssl = true
enable-connector-ssl = true
```

To display additional information in the report, use the `--extra` (or `-x`) option. For example:

```
shell> tpm report --extra --report 1

TUNGSTEN SECURITY REPORT as of Thu May  5 20:37:17 UTC 2022

----------------------------------------------------------------------
>> Comm Channel 1:  Application to Connector <<
----------------------------------------------------------------------

=> CHANNEL INFORMATION:
Connector Bridge Mode: OFF
Application-to-Connector SSL is Enabled (SSL.IN=true, Proxy Mode)

=> TPM OPTIONS:
connector-ssl-capable = true
enable-connector-client-ssl = true
enable-connector-ssl = true
```

To attempt to gather the information via the API, use `--api` on the command line. The tpm report command will fall back to the CLI tools if an API method is unavailable for a given report. For example:

```
shell> tpm report -x --report 1 --api

TUNGSTEN SECURITY REPORT as of Thu May  5 20:43:28 UTC 2022

----------------------------------------------------------------------
>> Comm Channel 1:  Application to Connector <<
----------------------------------------------------------------------

=> CHANNEL INFORMATION:
APIv2 Connector is SSL Capable: true
APIv2 Connector Requires SSL: true
APIv2 Connector Bridge Mode: false
WARN: No APIv2 solution available for Channel 1: Application to Connector - falling back to CLI
Application-to-Connector SSL is Enabled (SSL.IN=true, Proxy Mode)

=> TPM OPTIONS:
connector-ssl-capable = true
enable-connector-client-ssl = true
enable-connector-ssl = true
```

You may need to specify `--user` and `--password` for API authentication if not configured via tpm.

If both `--user` and `--password` are defined, tpm report will use them. If either or both `--user` and `--password` are missing, tpm report will attempt to derive the values from the configuration.

If you wish to output the report in machine-readable JSON-formatted text with other output suppressed, simply add the --json option. For example:

```
shell> tpm report -x --report 1 --api --json
{
   "1" : {
      "channelinformation" : [
         "APIv2 Connector is SSL Capable: true",
         "APIv2 Connector Requires SSL: true",
         "APIv2 Connector Bridge Mode: false",
         "WARN: No APIv2 solution available for Channel 1: Application to Connector - falling back to CLI",
         "Application-to-Connector SSL is Enabled (SSL.IN=true, Proxy Mode)"
      ],
      "metadata" : {
         "description" : "Communications from the client application to the Connector port",
         "protocol" : "mysql",
         "section" : "Application to Connector",
         "sslCapable" : 1
      },
      "tpmoptions" : {
         "connector-ssl-capable" : "true",
         "enable-connector-client-ssl" : "true",
         "enable-connector-ssl" : "true"
      }
   }
}
```

To simply display all reports, use:

```
shell> tpm report
shell> tpm report -x
shell> tpm report --json
shell> tpm report -x --json
```

Arguments:

Table 9.17. tpm report Common Options

| Option | Description |
|---|---|
| `--api` | Use the v2 API REST interface instead of the command line when possible |
| `--debug`, `-d` | |
| `--extra`, `-x` | Provide additional details in the reports |
| `--help`, `-h` | |
| `--info`, `-i` | |
| `--json` | Display report as JSON, all other output will be suppressed |
| `--list`, `-l` | List reports by number |
| `--password`, `-p` | Use to specify the API auth password (default: not defined) |
| `--path` | Use to supply full path to replicator executables |
| `--ports` | When available, display the hostname and listener ports |
| `--quiet`, `-q` | |
| `--report`, `--filter`, `-r` | Limit display to the specified report number(s); Use a comma-separated numeric list with no spaces to specify multiple reports. |
| `--ssl`, `-security` | Display current security settings and values (default behavior when no topic is specified) |
| `--test`, `-t` | |
| `--thl` | Use to supply full path to thl executable (Ignores --path) |
| `--trepctl` | Use to supply full path of trepctl executable (Ignore --path) |
| `--user`, `-u` | Use to specify API auth User (default: not defined) |
| `--verbose`, `-v` | |

## 9.5.16. tpm reset Command

This command will clear the current state for all Tungsten services:

- Management metadata
- Replication metadata
- THL files
- Relay log files
- Replication position

If you run the command from an installed directory, it will only apply to the current server. If you run it from a staging directory, it will apply to all servers unless you specify the `--hosts` [409] option.

```
shell> {STAGING_DIR}/tools/tpm reset
or
shell> tpm reset
```

## 9.5.17. tpm reset-thl Command

This command will clear the current replication state for the Tungsten Replicator:

- THL files

- Relay log files
- Replication position

If you run the command from an installed directory, it will only apply to the current server. If you run it from a staging directory, it will apply to all servers unless you specify the `--hosts` [409] option.

```
shell> {STAGING_DIR}/tools/tpm reset-thl
or
shell> tpm reset-thl
```

## 9.5.18. tpm reverse Command

The tpm reverse command will show you the commands required to rebuild the configuration for the current directory. This is useful for doing an upgrade or when copying the deployment to another server.

```
shell> tpm reverse
# Defaults for all data services and hosts
tools/tpm configure defaults \
--application-password=secret \
--application-port=3306 \
--application-user=app \
--replication-password=secret \
--replication-port=13306 \
--replication-user=tungsten \
--start-and-report=true \
--user=tungsten
# Options for the alpha data service
tools/tpm configure alpha \
--connectors=host1,host2,host3 \
--master=host1 \
--members=host1,host2,host3
```

The tpm reverse command supports the following arguments:

- `--public`

  Hide passwords in the command output

- `--ini-format`

  Display output in ini format for use in `/etc/tungsten/tungsten.ini` and similar configuration files

## 9.5.19. tpm uninstall Command

The tpm uninstall command is used to remove the installation.

> **Warning**
>
> The uninstall command must be used with care. This is a destructive command and irreversible.

To uninstall the software, you need to issue the following command from the installed software staging directory on every host for INI installs, or from the staging host only for Staging Installs. Running the command on the staging hosts installed via the staging method, will cascade through all nodes in the topology.

```
shell> {STAGING_DIR}/tools/tpm uninstall --i-am-sure
```

## 9.5.20. tpm update Command

The tpm update command is used when applying configuration changes or upgrading to a new version. The process is designed to be simple and maintain availability of all services. The actual process will be performed as described in Section 9.2, “Processing Installs and Upgrades”. The behavior of tpm update is dependent on two factors.

1. Are you upgrading to a new version or applying configuration changes to the current version?
2. The installation method used during deployment.

> **Note**
>
> Check the output of tpm query staging to determine which method your current installation uses. The output for an installation from a staging directory will start with `# Installed from tungsten@staging-host:/opt/continuent/software/tung-`

> sten-replicator-7.1.2-81. An installation based on an INI file may include this line but there will be an /etc/tungsten/tungsten.ini file on each node.

Upgrading to a new version

If a staging directory was used; see Section 9.3.6, “Upgrades from a Staging Directory”.

If an INI file was used; see Section 9.4.3, “Upgrades with an INI File”

Applying configuration changes to the current version

If a staging directory was used; see Section 9.3.7, “Configuration Changes from a Staging Directory”.

If an INI file was used; see Section 9.4.4, “Configuration Changes with an INI file”.

Special Considerations for the Connector

The tpm command will use connector graceful-stop 30 followed by connector start when upgrading versions. If that command fails then a regular connector stop is run.

This behavior is also applied when using tools/tpm update --replace-release.

The tpm command will use connector reconfigure when changing connector settings without a version upgrade.

The use of connector reconfigure is disabled for the following:

```
--application-port
--application-readonly-port
--router-gateway-port
--router-jmx-port
--conn-java-mem-size
```

If connector reconfigure can't be used, connector graceful-stop 30 and connector start are used.

## 9.5.21. tpm validate Command

The tpm validate command validates the current configuration before installation. The validation checks all prerequisites that apply before an installation, and assumes that the configured hosts are currently not configured for any Tungsten services, and no Tungsten services are currently running.

```
shell> {STAGING_DIR}/tools/tpm validate
.........
...
#####################################################################
# Validation failed
#####################################################################
...
```

The command can be run after performing a tpm configure and before a tpm install to ensure that any prerequisite or configuration issues are addressed before installation occurs.

## 9.5.22. tpm validate-update Command

The tpm validate-update command checks whether the configured hosts are ready to be updated. By checking the prerequisites and configuration of the dataserver and hosts, the same checks as made by tpm during a tpm install operation. Since there may have been changes to the requirements or required configuration, this check can be useful before attempting an update.

Using tpm validate-update is different from tpm validate in that it checks the environment based on the updated configuration, including the status of any existing services.

```
shell> {STAGING_DIR}/tools/tpm validate-update
....
WARN  >> host1 >> The process limit is set to 7812, we suggest a value»
    of at least 8096. Add 'tungsten       -    nproc  8096' to your »
    /etc/security/limits.conf and restart Tungsten processes. (ProcessLimitCheck)

WARN  >> host2 >> The process limit is set to 7812, we suggest a value»
     of at least 8096. Add 'tungsten       -    nproc  8096' to your »
    /etc/security/limits.conf and restart Tungsten processes. (ProcessLimitCheck)

WARN  >> host3 >> The process limit is set to 7812, we suggest a value »
    of at least 8096. Add 'tungsten       -    nproc  8096' to your »
    /etc/security/limits.conf and restart Tungsten processes. (ProcessLimitCheck)
.WARN  >> host3 >> MyISAM tables exist within this instance - These »
```

```
    tables are not crash safe and may lead to data loss in a failover »
    (MySQLMyISAMCheck)


NOTE  >> Command successfully completed
```

Any problems noted should be addressed before you perform the update using tpm update.

# 9.6. tpm Common Options

tpm accepts these options along with those in Section 9.8, "tpm Configuration Options".

- On the command-line, using a double-dash prefix, i.e. `--skip-validation-check=MySQLConnectorPermissionsCheck` [369]
- In an INI file, without the double-dash prefix, i.e. `skip-validation-check=MySQLConnectorPermissionsCheck` [369]

Table 9.18. tpm Common Options

| CmdLine Option | INI File Option | Description |
|---|---|---|
| `--enable-validation-check` [367] | `enable-validation-check` [367] | Enable a specific validation check, overriding any configured skipped checks |
| `--enable-validation-warnings` [367] | `enable-validation-warnings` [367] | Enable a specific validation warning, overriding any configured skipped warning |
| `--ini` [368] | `ini` [368] | Specify the location of the directory where INI files will be located, or specify a specific filename |
| `--net-ssh-option` [368] | `net-ssh-option` [368] | Set the Net::SSH option for remote system calls |
| `--property` [368], `--property=key+=value` [368], `--property=key=value` [368], `--property=key~=/match/replace/` [368] | `property` [368], `property=key+=value` [368], `property=key=value` [368], `property=key~=/match/replace/` [368] | Modify specific property values for the key in any file that the configure script touches. |
| `--remove-property` [368] | `remove-property` [368] | Remove the setting for a previously configured property |
| `--replace-release` [368] | `replace-release` [368] | Used when upgrading or making configuration changes. This option will rebuild the entire release and recreate all metadata configuration. |
| `--skip-validation-check` [369] | `skip-validation-check` [369] | Do not run the specified validation check. |
| `--skip-validation-warnings` [369] | `skip-validation-warnings` [369] | Do not display warnings for the specified validation check. |

– `--enable-validation-check` [367]

| | |
|---|---|
| Option | `--enable-validation-check` [367] |
| Config File Options | `enable-validation-check` [367] |
| Description | Enable a specific validation check, overriding any configured skipped checks |
| Value Type | string |

The `--enable-validation-check` [367] will specifically enable a given validation check if the check had previously been set it be ignored in a previous invocation of the configuration through tpm. If a check fails, installation is canceled.

Setting both `--skip-validation-check` [369] and `--enable-validation-check` [367] is equivalent to explicitly disabling the specified check.

– `--enable-validation-warnings` [367]

| | |
|---|---|
| Option | `--enable-validation-warnings` [367] |
| Config File Options | `enable-validation-warnings` [367] |
| Description | Enable a specific validation warning, overriding any configured skipped warning |
| Value Type | string |

The `--enable-validation-warnings` [367] will specifically enable a given validation warning check if the check had previously been set it be ignored in a previous invocation of the configuration through tpm.

Setting both `--skip-validation-warnings` [369] and `--enable-validation-warnings` [367] is equivalent to explicitly disabling the specified check.

– --ini [368]

| Option | --ini [368] |
|---|---|
| Config File Options | ini [368] |
| Description | Specify the location of the directory where INI files will be located, or specify a specific filename |
| Value Type | string |
| Default | /etc/tungsten/tungsten.ini |

Specifies an alternative location, or file, for the INI files from the default.

– --net-ssh-option [368]

| Option | --net-ssh-option [368] |
|---|---|
| Config File Options | net-ssh-option [368] |
| Description | Set the Net::SSH option for remote system calls |
| Value Type | string |

Enables you to set a specific Net::SSH option. For example:

```
shell> tpm update ... --net-ssh-option=compression=zlib
```

– --property [368]

| Option | --property [368] |
|---|---|
| Aliases | --property=key+=value [368], --property=key=value [368], --property=key~=/match/replace/ [368] |
| Config File Options | property [368], property=key+=value [368], property=key=value [368], property=key~=/match/replace/ [368] |
| Description | Modify specific property values for the key in any file that the configure script touches. |
| Value Type | string |

The --property [368] option enables you to explicitly set property values in the target files. A number of different models are supported:

- key=value

  Set the property defined by key to the specified value without evaluating any template values or other rules.

- key+=value

  Add the value to the property defined by key. Template values and other options append their settings to the end of the specified property.

- key~=/match/replace/

  Evaluate any template values and other settings, and then perform the specified Ruby regex operation to the property defined by key. For example --property=replicator.key~=/(.*)/somevalue,\1/ will prepend somevalue before the template value for replicator.key.

– --remove-property [368]

| Option | --remove-property [368] |
|---|---|
| Config File Options | remove-property [368] |
| Description | Remove the setting for a previously configured property |
| Value Type | string |

Remove a previous explicit property setting. For example:

```
shell> tpm configure --remove-property=replicator.filter.pkey.addPkeyToInserts
```

– --replace-release [368]

| Option | --replace-release [368] |
|---|---|
| Config File Options | replace-release [368] |

| | |
|---|---|
| Description | Used when upgrading or making configuration changes. This option will rebuild the entire release and recreate all metadata configuration. |
| Value Type | boolean |

This property can only be used with tools/tpm update and can only be executed from within the software staging tree.

This option is highly recommended when upgrading between versions as it will ensure all metadata is correctly rebuilt for the version being installed.

It is not necessary when applying config changes, however it can be useful if properties are not being applied correctly, or if underlying metadata has become corrupt.

– `--skip-validation-check` [369]

| | |
|---|---|
| Option | `--skip-validation-check` [369] |
| Config File Options | `skip-validation-check` [369] |
| Description | Do not run the specified validation check. |
| Value Type | string |

The `--skip-validation-check` [369] disables a given validation check. If any validation check fails, the installation, validation or configuration will automatically stop.

### Warning

Using this option enables you to bypass the specified check, although skipping a check may lead to an invalid or non-working configuration.

You can identify a given check if an error or warning has been raised during configuration. For example, the default table type check:

```
...
ERROR >> centos >> The datasource root@centos:3306 (WITH PASSWORD) »
 uses MyISAM as the default storage engine (MySQLDefaultTableTypeCheck)
...
```

The check in this case is `MySQLDefaultTableTypeCheck` [380], and could be ignored using `--skip-validation-check=MySQLDefaultTableTypeCheck` [369].

Setting both `--skip-validation-check` [369] and `--enable-validation-check` [367] is equivalent to explicitly disabling the specified check.

– `--skip-validation-warnings` [369]

| | |
|---|---|
| Option | `--skip-validation-warnings` [369] |
| Config File Options | `skip-validation-warnings` [369] |
| Description | Do not display warnings for the specified validation check. |
| Value Type | string |

The `--skip-validation-warnings` [369] disables a given validation check.

You can identify a given check by examining the warnings generated during configuration. For example, the Linux swappiness warning:

```
...
WARN >> centos >> Linux swappiness is currently set to 60, on restart it will be 60, »
 consider setting this to 10 or under to avoid swapping. (SwappinessCheck)
...
```

The check in this case is `MySQLDefaultTableTypeCheck` [380], and could be ignored using `--skip-validation-warnings=SwappinessCheck` [369].

Setting both `--skip-validation-warnings` [369] and `--enable-validation-warnings` [367] is equivalent to explicitly disabling the specified warning.

## 9.7. tpm Validation Checks

During configuration and installation, tpm runs a number of configuration, operating system, datasource, and other validation checks to ensure that the correct environment, prerequisites and other settings will produce a valid, working, configuration.

All relevant checks are executed automatically unless specifically ignored (warnings) or disabled (checks) using the corresponding `--skip-validation-warnings` [369] or `--skip-validation-check` [369] options.

Table 9.19. tpm Validation Checks

| Option | Description |
| --- | --- |
| BackupDirectoryWriteableCheck [373] | Checks that the configured backup directory is writeable |
| BackupDumpDirectoryWriteableCheck [373] | Checks the backup temp directory is writeable |
| BackupScriptAvailableCheck [373] | Checks that the configured backup script exists and can be executed |
| ClusterDiagnosticCheck [373] | |
| ClusterStatusCheck [373] | |
| CommitDirectoryCheck [373] | |
| ConfigurationStorageDirectoryCheck [374] | |
| ConfigureValidationCheck [374] | |
| ConfiguredDirectoryCheck [374] | |
| ConflictingReplicationServiceTHLPortsCheck [374] | |
| ConnectorChecks [374] | Ensures that the configured connector selection is valid |
| ConnectorDBVersionCheck [374] | |
| ConnectorListenerAddressCheck [374] | |
| ConnectorRWROAddressesCheck [374] | Ensure the RW and RO addresses are different |
| ConnectorSmartScaleAllowedCheck [374] | Confirms whether SmartScale is valid within the current configured parameters |
| ConnectorUserCheck [374] | |
| ConsistentReplicationCredentialsCheck [375] | |
| CurrentCommandCoordinatorCheck [375] | |
| CurrentConnectorCheck [375] | |
| CurrentReleaseDirectoryIsSymlink [375] | |
| CurrentTopologyCheck [375] | |
| CurrentVersionCheck [375] | |
| DatasourceBootScriptCheck [375] | |
| DifferentMasterSlaveCheck [375] | |
| DirectOracleServiceSIDCheck [375] | |
| EncryptionCheck [375] | |
| EncryptionKeystoreCheck [375] | |
| FileValidationCheck [376] | |
| FirewallCheck [376] | |
| GlobalHostAddressesCheck [376] | |
| GlobalHostOracleLibrariesFoundCheck [376] | |
| GlobalMatchingPingMethodCheck [376] | |
| GlobalRestartComponentsCheck [376] | |
| GroupValidationCheck [376] | |
| HdfsValidationCheck [376] | |
| HostLicensesCheck [376] | |
| HostOracleLibrariesFoundCheck [376] | |
| HostReplicatorServiceRunningCheck [376] | |
| HostSkippedChecks [377] | |
| HostnameCheck [377] | |
| HostsFileCheck [377] | |
| InstallServicesCheck [377] | |
| InstallationScriptCheck [377] | |

| Option | Description |
| --- | --- |
| InstallerMasterSlaveCheck [377] | Checks whether a Primary host has been defined for the configured service. |
| InstallingOverExistingInstallation [377] | |
| JavaUserTimezoneCheck [377] | |
| JavaVersionCheck [377] | |
| KeystoresCheck [377] | |
| KeystoresToCommitCheck [377] | |
| ManagerActiveWitnessConversionCheck [378] | |
| ManagerChecks [378] | |
| ManagerHeapThresholdCheck [378] | |
| ManagerListenerAddressCheck [378] | |
| ManagerPingMethodCheck [378] | |
| ManagerWitnessAvailableCheck [378] | |
| ManagerWitnessNeededCheck [378] | |
| MatchingHomeDirectoryCheck [378] | |
| MissingReplicationServiceConfigurationCheck [378] | |
| ModifiedConfigurationFilesCheck [378] | |
| MySQLAllowIntensiveChecks [378] | Enables searching MySQL INFORMATION_SCHEMA for validation checks |
| MySQLApplierLogsCheck [379] | |
| MySQLApplierPortCheck [379] | |
| MySQLApplierServerIDCheck [379] | |
| MySQLAvailableCheck [379] | Checks if MySQL is installed |
| MySQLBinaryLogsEnabledCheck [379] | Checks that binary logging has been enabled on MySQL |
| MySQLBinlogDoDbCheck [379] | |
| MySQLClientCheck [379] | Checks whether the MySQL client command tool is available |
| MySQLConfigFileCheck [379] | Checks the existence of a MySQL configuration file |
| MySQLConnectorBridgeModePermissionsCheck [379] | |
| MySQLConnectorPermissionsCheck [379] | |
| MySQLDefaultTableTypeCheck [380] | Checks the default table type for MySQL |
| MySQLDumpCheck [380] | Checks that the mysqldump command version matches the installed MySQL |
| MySQLGeneratedColumnCheck [380] | Checks whether MySQL virtual/generated columns are defined |
| MySQLInnoDBEnabledCheck [380] | |
| MySQLJsonDataTypeCheck [380] | |
| MySQLLoadDataInfilePermissionsCheck [380] | |
| MySQLLoginCheck [380] | Checks whether Tungsten Cluster can connect to MySQL using the configured credentials |
| MySQLMyISAMCheck [380] | Checks for the existence of MyISAM tables |
| MySQLNoMySQLReplicationCheck [381] | |
| MySQLPasswordSettingCheck [381] | |
| MySQLPermissionsCheck [381] | |
| MySQLReadableLogsCheck [381] | |
| MySQLSettingsCheck [381] | |
| MySQLSuperReadOnlyCheck [381] | Checks whether super_read_only has been enabled on MySQL |
| MySQLTriggerCheck [381] | |

| Option | Description |
|---|---|
| MySQLUnsupportedDataTypesCheck [381] | |
| MysqlConnectorCheck [381] | |
| MysqldumpAvailableCheck [382] | |
| MysqldumpSettingsCheck [382] | |
| NewDirectoryRequiredCheck [382] | |
| NtpdRunningCheck [382] | |
| OSCheck [382] | |
| OldServicesRunningCheck [382] | |
| OpenFilesLimitCheck [382] | |
| OpensslLibraryCheck [382] | |
| OracleLoginCheck [382] | |
| OraclePermissionsCheck [382] | |
| OracleRedoReaderMinerDirectoryCheck [382] | |
| OracleServiceSIDCheck [383] | |
| OracleVersionCheck [383] | |
| PGAvailableCheck [383] | |
| ParallelReplicationCheck [383] | |
| ParallelReplicationCountCheck [383] | |
| PgControlAvailableCheck [383] | |
| PgStandbyAvailableCheck [383] | |
| PgdumpAvailableCheck [383] | |
| PgdumpallAvailableCheck [383] | |
| PingSyntaxCheck [383] | |
| PortAvailabilityCheck [383] | |
| ProfileScriptCheck [384] | |
| RMIListenerAddressCheck [384] | |
| RelayDirectoryWriteableCheck [384] | Checks that the relay log directory can be written to |
| ReplicatorChecks [384] | |
| RestartComponentsCheck [384] | |
| RouterAffinityCheck [384] | |
| RouterBridgeModeDefaultCheck [384] | |
| RouterDelayBeforeOfflineCheck [384] | |
| RouterKeepAliveTimeoutCheck [384] | |
| RowBasedBinaryLoggingCheck [384] | Checks that Row-based binary logging has been enabled for heterogeneous deployments |
| RsyncAvailableCheck [385] | |
| RubyVersionCheck [385] | |
| SSHLoginCheck [385] | Checks connectivity to other hosts over SSH |
| ServiceTransferredLogStorageCheck [385] | |
| StartingStoppedServices [385] | |
| SudoCheck [385] | |
| SwappinessCheck [385] | Checks the swappiness OS configuration is within a recommended range |
| THLDirectoryWriteableCheck [385] | |
| THLListenerAddressCheck [386] | |

| Option | Description |
| --- | --- |
| THLSchemaChangeCheck [386] | Ensures that the existing THL format is compatible with the new release |
| THLStorageCheck [386] | Confirms the THL storage directory exists, is empty and writeable |
| THLStorageChecksum [386] | |
| TargetDirectoryDoesNotExist [386] | |
| TransferredLogStorageCheck [386] | |
| UpgradeSameProductCheck [386] | Ensures that the same product is being updated |
| VIPEnabledHostAllowsRootCommands [386] | |
| VIPEnabledHostArpPath [386] | |
| VIPEnabledHostIfconfigPath [386] | |
| VerticaUserGroupsCheck [387] | Checks that the Vertica user has the correct OS group membership |
| WhichAvailableCheck [387] | Checks the existence of a working which command |
| WriteableHomeDirectoryCheck [387] | Ensures the home directory can be written to |
| WriteableTempDirectoryCheck [387] | Ensures the temporary directory can be written to |
| XtrabackupAvailableCheck [387] | |
| XtrabackupDirectoryWriteableCheck [387] | |
| XtrabackupSettingsCheck [387] | |

– BackupDirectoryWriteableCheck [373]

| Option | BackupDirectoryWriteableCheck [373] |
| --- | --- |
| Description | Checks that the configured backup directory is writeable |

Confirms that the directory defined in --backup-dir directory exists and can be written to.

– BackupDumpDirectoryWriteableCheck [373]

| Option | BackupDumpDirectoryWriteableCheck [373] |
| --- | --- |
| Description | Checks the backup temp directory is writeable |

Confirms that the directory defined in --backup-dump-dir directory exists and can be written to.

– BackupScriptAvailableCheck [373]

| Option | BackupScriptAvailableCheck [373] |
| --- | --- |
| Description | Checks that the configured backup script exists and can be executed |

Confirms that the script defined in --backup-script [397] exists and is executable.

– ClusterDiagnosticCheck [373]

| Option | ClusterDiagnosticCheck [373] |
| --- | --- |
| Description | |

– ClusterStatusCheck [373]

| Option | ClusterStatusCheck [373] |
| --- | --- |
| Description | |

– CommitDirectoryCheck [373]

| Option | CommitDirectoryCheck [373] |
| --- | --- |

| | |
|---|---|
| Description | |

– ConfigurationStorageDirectoryCheck [374]

| | |
|---|---|
| Option | ConfigurationStorageDirectoryCheck [374] |
| Description | |

– ConfigureValidationCheck [374]

| | |
|---|---|
| Option | ConfigureValidationCheck [374] |
| Description | |

– ConfiguredDirectoryCheck [374]

| | |
|---|---|
| Option | ConfiguredDirectoryCheck [374] |
| Description | |

– ConflictingReplicationServiceTHLPortsCheck [374]

| | |
|---|---|
| Option | ConflictingReplicationServiceTHLPortsCheck [374] |
| Description | |

– ConnectorChecks [374]

| | |
|---|---|
| Option | ConnectorChecks [374] |
| Description | Ensures that the configured connector selection is valid |

Checks that the list of connectors and the corresponding list of data services is valid.

– ConnectorDBVersionCheck [374]

| | |
|---|---|
| Option | ConnectorDBVersionCheck [374] |
| Description | |

– ConnectorListenerAddressCheck [374]

| | |
|---|---|
| Option | ConnectorListenerAddressCheck [374] |
| Description | |

– ConnectorRWROAddressesCheck [374]

| | |
|---|---|
| Option | ConnectorRWROAddressesCheck [374] |
| Description | Ensure the RW and RO addresses are different |

For environments where the connector has been configured to use different hosts and ports for RW and RO operations, ensure that the settings are in fact different.

– ConnectorSmartScaleAllowedCheck [374]

| | |
|---|---|
| Option | ConnectorSmartScaleAllowedCheck [374] |
| Description | Confirms whether SmartScale is valid within the current configured parameters |

Checks that both SmartScale and Read/Write splitting have been enabled.

– ConnectorUserCheck [374]

| | |
|---|---|
| Option | ConnectorUserCheck [374] |

| | |
|---|---|
| Description | |

– ConsistentReplicationCredentialsCheck [375]

| | |
|---|---|
| Option | ConsistentReplicationCredentialsCheck [375] |
| Description | |

– CurrentCommandCoordinatorCheck [375]

| | |
|---|---|
| Option | CurrentCommandCoordinatorCheck [375] |
| Description | |

– CurrentConnectorCheck [375]

| | |
|---|---|
| Option | CurrentConnectorCheck [375] |
| Description | |

– CurrentReleaseDirectoryIsSymlink [375]

| | |
|---|---|
| Option | CurrentReleaseDirectoryIsSymlink [375] |
| Description | |

– CurrentTopologyCheck [375]

| | |
|---|---|
| Option | CurrentTopologyCheck [375] |
| Description | |

– CurrentVersionCheck [375]

| | |
|---|---|
| Option | CurrentVersionCheck [375] |
| Description | |

– DatasourceBootScriptCheck [375]

| | |
|---|---|
| Option | DatasourceBootScriptCheck [375] |
| Description | |

– DifferentMasterSlaveCheck [375]

| | |
|---|---|
| Option | DifferentMasterSlaveCheck [375] |
| Description | |

– DirectOracleServiceSIDCheck [375]

| | |
|---|---|
| Option | DirectOracleServiceSIDCheck [375] |
| Description | |

– EncryptionCheck [375]

| | |
|---|---|
| Option | EncryptionCheck [375] |
| Description | |

– EncryptionKeystoreCheck [375]

| | |
|---|---|
| Option | EncryptionKeystoreCheck [375] |

| | |
|---|---|
| Description | |

– FileValidationCheck [376]

| | |
|---|---|
| Option | FileValidationCheck [376] |
| Description | |

– FirewallCheck [376]

| | |
|---|---|
| Option | FirewallCheck [376] |
| Description | |

– GlobalHostAddressesCheck [376]

| | |
|---|---|
| Option | GlobalHostAddressesCheck [376] |
| Description | |

– GlobalHostOracleLibrariesFoundCheck [376]

| | |
|---|---|
| Option | GlobalHostOracleLibrariesFoundCheck [376] |
| Description | |

– GlobalMatchingPingMethodCheck [376]

| | |
|---|---|
| Option | GlobalMatchingPingMethodCheck [376] |
| Description | |

– GlobalRestartComponentsCheck [376]

| | |
|---|---|
| Option | GlobalRestartComponentsCheck [376] |
| Description | |

– GroupValidationCheck [376]

| | |
|---|---|
| Option | GroupValidationCheck [376] |
| Description | |

– HdfsValidationCheck [376]

| | |
|---|---|
| Option | HdfsValidationCheck [376] |
| Description | |

– HostLicensesCheck [376]

| | |
|---|---|
| Option | HostLicensesCheck [376] |
| Description | |

– HostOracleLibrariesFoundCheck [376]

| | |
|---|---|
| Option | HostOracleLibrariesFoundCheck [376] |
| Description | |

– HostReplicatorServiceRunningCheck [376]

| | |
|---|---|
| Option | HostReplicatorServiceRunningCheck [376] |

| | |
|---|---|
| Description | |

– HostSkippedChecks [377]

| | |
|---|---|
| Option | HostSkippedChecks [377] |
| Description | |

– HostnameCheck [377]

| | |
|---|---|
| Option | HostnameCheck [377] |
| Description | |

– HostsFileCheck [377]

| | |
|---|---|
| Option | HostsFileCheck [377] |
| Description | |

– InstallServicesCheck [377]

| | |
|---|---|
| Option | InstallServicesCheck [377] |
| Description | |

– InstallationScriptCheck [377]

| | |
|---|---|
| Option | InstallationScriptCheck [377] |
| Description | |

– InstallerMasterSlaveCheck [377]

| | |
|---|---|
| Option | InstallerMasterSlaveCheck [377] |
| Description | Checks whether a Primary host has been defined for the configured service. |

– InstallingOverExistingInstallation [377]

| | |
|---|---|
| Option | InstallingOverExistingInstallation [377] |
| Description | |

– JavaUserTimezoneCheck [377]

| | |
|---|---|
| Option | JavaUserTimezoneCheck [377] |
| Description | |

– JavaVersionCheck [377]

| | |
|---|---|
| Option | JavaVersionCheck [377] |
| Description | |

– KeystoresCheck [377]

| | |
|---|---|
| Option | KeystoresCheck [377] |
| Description | |

– KeystoresToCommitCheck [377]

| | |
|---|---|
| Option | KeystoresToCommitCheck [377] |

| | |
|---|---|
| Description | |

– ManagerActiveWitnessConversionCheck [378]

| | |
|---|---|
| Option | ManagerActiveWitnessConversionCheck [378] |
| Description | |

– ManagerChecks [378]

| | |
|---|---|
| Option | ManagerChecks [378] |
| Description | |

– ManagerHeapThresholdCheck [378]

| | |
|---|---|
| Option | ManagerHeapThresholdCheck [378] |
| Description | |

– ManagerListenerAddressCheck [378]

| | |
|---|---|
| Option | ManagerListenerAddressCheck [378] |
| Description | |

– ManagerPingMethodCheck [378]

| | |
|---|---|
| Option | ManagerPingMethodCheck [378] |
| Description | |

– ManagerWitnessAvailableCheck [378]

| | |
|---|---|
| Option | ManagerWitnessAvailableCheck [378] |
| Description | |

– ManagerWitnessNeededCheck [378]

| | |
|---|---|
| Option | ManagerWitnessNeededCheck [378] |
| Description | |

– MatchingHomeDirectoryCheck [378]

| | |
|---|---|
| Option | MatchingHomeDirectoryCheck [378] |
| Description | |

– MissingReplicationServiceConfigurationCheck [378]

| | |
|---|---|
| Option | MissingReplicationServiceConfigurationCheck [378] |
| Description | |

– ModifiedConfigurationFilesCheck [378]

| | |
|---|---|
| Option | ModifiedConfigurationFilesCheck [378] |
| Description | |

– MySQLAllowIntensiveChecks [378]

| | |
|---|---|
| Option | MySQLAllowIntensiveChecks [378] |

| | |
|---|---|
| Description | Enables searching MySQL INFORMATION_SCHEMA for validation checks |

Enables tpm to make use of the MySQL `INFORMATION_SCHEMA` to perform various validation checks. These include, but are not limited to:

- Tables not configured to use transactional tables
- Unsupported datatypes in MySQL tables

– `MySQLApplierLogsCheck` [379]

| | |
|---|---|
| Option | `MySQLApplierLogsCheck` [379] |
| Description | |

– `MySQLApplierPortCheck` [379]

| | |
|---|---|
| Option | `MySQLApplierPortCheck` [379] |
| Description | |

– `MySQLApplierServerIDCheck` [379]

| | |
|---|---|
| Option | `MySQLApplierServerIDCheck` [379] |
| Description | |

– `MySQLAvailableCheck` [379]

| | |
|---|---|
| Option | `MySQLAvailableCheck` [379] |
| Description | Checks if MySQL is installed |

– `MySQLBinaryLogsEnabledCheck` [379]

| | |
|---|---|
| Option | `MySQLBinaryLogsEnabledCheck` [379] |
| Description | Checks that binary logging has been enabled on MySQL |

Examines the `log_bin` variable has been defined within the running MySQL server. Binary logging must be enabled for replication to work.

– `MySQLBinlogDoDbCheck` [379]

| | |
|---|---|
| Option | `MySQLBinlogDoDbCheck` [379] |
| Description | |

– `MySQLClientCheck` [379]

| | |
|---|---|
| Option | `MySQLClientCheck` [379] |
| Description | Checks whether the MySQL client command tool is available |

– `MySQLConfigFileCheck` [379]

| | |
|---|---|
| Option | `MySQLConfigFileCheck` [379] |
| Description | Checks the existence of a MySQL configuration file |

– `MySQLConnectorBridgeModePermissionsCheck` [379]

| | |
|---|---|
| Option | `MySQLConnectorBridgeModePermissionsCheck` [379] |
| Description | |

– `MySQLConnectorPermissionsCheck` [379]

| | |
|---|---|
| Option | `MySQLConnectorPermissionsCheck` [379] |

| | |
|---|---|
| Description | |

– MySQLDefaultTableTypeCheck [380]

| | |
|---|---|
| Option | MySQLDefaultTableTypeCheck [380] |
| Description | Checks the default table type for MySQL |

Checks that the default table type configured for MySQL is a compatible transactional storage engine such as InnoDB

– MySQLDumpCheck [380]

| | |
|---|---|
| Option | MySQLDumpCheck [380] |
| Description | Checks that the mysqldump command version matches the installed MySQL |

Checks whether the mysqldump command within the configured PATH matches the version of MySQL being configured as a source or target. A mismatch could indicate that multiple MySQL versions are installed.

A mismatch could create invalid or corrupt backups. Either correct your PATH or use --preferred-path [416] to point to the correct MySQL installation.

– MySQLGeneratedColumnCheck [380]

| | |
|---|---|
| Option | MySQLGeneratedColumnCheck [380] |
| Description | Checks whether MySQL virtual/generated columns are defined |

Checks, whether any tables contain generated or virtual columns. The test is only executed on MySQL 5.7 and only if --mysql-allow-intensive-checks [413] has been enabled.

– MySQLInnoDBEnabledCheck [380]

| | |
|---|---|
| Option | MySQLInnoDBEnabledCheck [380] |
| Description | |

– MySQLJsonDataTypeCheck [380]

| | |
|---|---|
| Option | MySQLJsonDataTypeCheck [380] |
| Description | |

Checks, whether any tables contain JSON columns. The test is only executed on MySQL 5.7 and only if --mysql-allow-intensive-checks [413] has been enabled.

– MySQLLoadDataInfilePermissionsCheck [380]

| | |
|---|---|
| Option | MySQLLoadDataInfilePermissionsCheck [380] |
| Description | |

– MySQLLoginCheck [380]

| | |
|---|---|
| Option | MySQLLoginCheck [380] |
| Description | Checks whether Tungsten Cluster can connect to MySQL using the configured credentials |

– MySQLMyISAMCheck [380]

| | |
|---|---|
| Option | MySQLMyISAMCheck [380] |
| Description | Checks for the existence of MyISAM tables |

Checks for the existence of MyISAM tables within the database. Use of MyISAM tables is not supported since MyISAM is not transactionally consistent. This can cause problems for both extraction and applying data.

In order to check for the existence of MyISAM tables, tpm uses two techniques:

- Looking for `.MYD` files within the MySQL directory, which are the files which contains MyISAM data. tpm must be able to read and see the contents of the MySQL data directory. If the configured user does not already have access, you can use the `--root-command-prefix=true` [407] option to grant root access to access the filesystem.
- Using the MySQL `INFORMATION_SCHEMA` to look for tables defined with the MyISAM engine. For this option to work, intensive checks must have been enabled using `--mysql-allow-intensive-checks` [413].

If neither of these methods is available, the check will fail and installation will stop.

– `MySQLNoMySQLReplicationCheck` [381]

| Option | `MySQLNoMySQLReplicationCheck` [381] |
|---|---|
| Description | |

– `MySQLPasswordSettingCheck` [381]

| Option | `MySQLPasswordSettingCheck` [381] |
|---|---|
| Description | |

– `MySQLPermissionsCheck` [381]

| Option | `MySQLPermissionsCheck` [381] |
|---|---|
| Description | |

– `MySQLReadableLogsCheck` [381]

| Option | `MySQLReadableLogsCheck` [381] |
|---|---|
| Description | |

– `MySQLSettingsCheck` [381]

| Option | `MySQLSettingsCheck` [381] |
|---|---|
| Description | |

– `MySQLSuperReadOnlyCheck` [381]

| Option | `MySQLSuperReadOnlyCheck` [381] |
|---|---|
| Description | Checks whether super_read_only has been enabled on MySQL |

Checks whether the `super_read_only` variable within MySQL has been enabled. If enabled, replication will not work. The check will test both the running server and the configuration file to determine whether the value has been enabled.

– `MySQLTriggerCheck` [381]

| Option | `MySQLTriggerCheck` [381] |
|---|---|
| Description | |

– `MySQLUnsupportedDataTypesCheck` [381]

| Option | `MySQLUnsupportedDataTypesCheck` [381] |
|---|---|
| Description | |

– `MysqlConnectorCheck` [381]

| Option | `MysqlConnectorCheck` [381] |
|---|---|

| | |
|---|---|
| Description | |

– MysqldumpAvailableCheck [382]

| | |
|---|---|
| Option | MysqldumpAvailableCheck [382] |
| Description | |

– MysqldumpSettingsCheck [382]

| | |
|---|---|
| Option | MysqldumpSettingsCheck [382] |
| Description | |

– NewDirectoryRequiredCheck [382]

| | |
|---|---|
| Option | NewDirectoryRequiredCheck [382] |
| Description | |

– NtpdRunningCheck [382]

| | |
|---|---|
| Option | NtpdRunningCheck [382] |
| Description | |

– OSCheck [382]

| | |
|---|---|
| Option | OSCheck [382] |
| Description | |

– OldServicesRunningCheck [382]

| | |
|---|---|
| Option | OldServicesRunningCheck [382] |
| Description | |

– OpenFilesLimitCheck [382]

| | |
|---|---|
| Option | OpenFilesLimitCheck [382] |
| Description | |

– OpensslLibraryCheck [382]

| | |
|---|---|
| Option | OpensslLibraryCheck [382] |
| Description | |

– OracleLoginCheck [382]

| | |
|---|---|
| Option | OracleLoginCheck [382] |
| Description | |

– OraclePermissionsCheck [382]

| | |
|---|---|
| Option | OraclePermissionsCheck [382] |
| Description | |

– OracleRedoReaderMinerDirectoryCheck [382]

| | |
|---|---|
| Option | OracleRedoReaderMinerDirectoryCheck [382] |

| | |
|---|---|
| Description | |

– OracleServiceSIDCheck [383]

| | |
|---|---|
| Option | OracleServiceSIDCheck [383] |
| Description | |

– OracleVersionCheck [383]

| | |
|---|---|
| Option | OracleVersionCheck [383] |
| Description | |

– PGAvailableCheck [383]

| | |
|---|---|
| Option | PGAvailableCheck [383] |
| Description | |

– ParallelReplicationCheck [383]

| | |
|---|---|
| Option | ParallelReplicationCheck [383] |
| Description | |

– ParallelReplicationCountCheck [383]

| | |
|---|---|
| Option | ParallelReplicationCountCheck [383] |
| Description | |

– PgControlAvailableCheck [383]

| | |
|---|---|
| Option | PgControlAvailableCheck [383] |
| Description | |

– PgStandbyAvailableCheck [383]

| | |
|---|---|
| Option | PgStandbyAvailableCheck [383] |
| Description | |

– PgdumpAvailableCheck [383]

| | |
|---|---|
| Option | PgdumpAvailableCheck [383] |
| Description | |

– PgdumpallAvailableCheck [383]

| | |
|---|---|
| Option | PgdumpallAvailableCheck [383] |
| Description | |

– PingSyntaxCheck [383]

| | |
|---|---|
| Option | PingSyntaxCheck [383] |
| Description | |

– PortAvailabilityCheck [383]

| | |
|---|---|
| Option | PortAvailabilityCheck [383] |

| | |
|---|---|
| Description | |

– ProfileScriptCheck [384]

| | |
|---|---|
| Option | ProfileScriptCheck [384] |
| Description | |

– RMIListenerAddressCheck [384]

| | |
|---|---|
| Option | RMIListenerAddressCheck [384] |
| Description | |

– RelayDirectoryWriteableCheck [384]

| | |
|---|---|
| Option | RelayDirectoryWriteableCheck [384] |
| Description | Checks that the relay log directory can be written to |

Confirms that the directory defined in --relay-log-dir directory exists and can be written to.

– ReplicatorChecks [384]

| | |
|---|---|
| Option | ReplicatorChecks [384] |
| Description | |

– RestartComponentsCheck [384]

| | |
|---|---|
| Option | RestartComponentsCheck [384] |
| Description | |

– RouterAffinityCheck [384]

| | |
|---|---|
| Option | RouterAffinityCheck [384] |
| Description | |

– RouterBridgeModeDefaultCheck [384]

| | |
|---|---|
| Option | RouterBridgeModeDefaultCheck [384] |
| Description | |

– RouterDelayBeforeOfflineCheck [384]

| | |
|---|---|
| Option | RouterDelayBeforeOfflineCheck [384] |
| Description | |

– RouterKeepAliveTimeoutCheck [384]

| | |
|---|---|
| Option | RouterKeepAliveTimeoutCheck [384] |
| Description | |

– RowBasedBinaryLoggingCheck [384]

| | |
|---|---|
| Option | RowBasedBinaryLoggingCheck [384] |
| Description | Checks that Row-based binary logging has been enabled for heterogeneous deployments |

For heterogeneous deployments, row-based binary logging must have been enabled. For all services where heterogeneous support has been enabled, for example due to --enable-heterogeneous-service [406] or --enable-batch-service [405], row-based logging within MySQL must have been switched on. The test looks for the value of binlog_format=ROW.

– RsyncAvailableCheck [385]

| | |
|---|---|
| Option | RsyncAvailableCheck [385] |
| Description | |

– RubyVersionCheck [385]

| | |
|---|---|
| Option | RubyVersionCheck [385] |
| Description | |

– SSHLoginCheck [385]

| | |
|---|---|
| Option | SSHLoginCheck [385] |
| Description | Checks connectivity to other hosts over SSH |

Checks to confirm the SSH logins to other hosts in the cluster work, without requiring a password, and without returning additional rows of information when directly, remotely, running a command.

In the event of the check failing, the following items should be checked:

- Confirm that it is possible to SSH to the remote site using the username provided, and without requiring a password. For example:

```
host1-shell> ssh tungsten@host2
Last login: Wed Aug 9 09:55:23 2017 from fe80::1042:8aee:61da:a20%en0
host2-shell>
```

- Remove any remote messages returned when the user logs in. This includes the output from the *Banner* argument within /etc/ssh/sshd_config, or text or files output by the users shell login script or profile.

- Ensure that your remote shell has not been configured to output text or a message when a logout is attempted, for example by using:

```
shell> trap "echo logout" 0
```

– ServiceTransferredLogStorageCheck [385]

| | |
|---|---|
| Option | ServiceTransferredLogStorageCheck [385] |
| Description | |

– StartingStoppedServices [385]

| | |
|---|---|
| Option | StartingStoppedServices [385] |
| Description | |

– SudoCheck [385]

| | |
|---|---|
| Option | SudoCheck [385] |
| Description | |

– SwappinessCheck [385]

| | |
|---|---|
| Option | SwappinessCheck [385] |
| Description | Checks the swappiness OS configuration is within a recommended range |

Checks whether the Linux swappiness parameter has been set to a value of 10 or less, both in the current setting and when the system reboots. A value greater than 10 may allow for running programs to be swapped out, which will affect the performance of the Tungsten Cluster when running. Change the value in sysctl.conf.

– THLDirectoryWriteableCheck [385]

| | |
|---|---|
| Option | THLDirectoryWriteableCheck [385] |

| | |
|---|---|
| Description | |

– THLListenerAddressCheck [386]

| | |
|---|---|
| Option | THLListenerAddressCheck [386] |
| Description | |

– THLSchemaChangeCheck [386]

| | |
|---|---|
| Option | THLSchemaChangeCheck [386] |
| Description | Ensures that the existing THL format is compatible with the new release |

Checks that the format of the current THL is compatible with the schema and format of the new software. A difference may mean that the THL needs to be reset before installation can continue.

– THLStorageCheck [386]

| | |
|---|---|
| Option | THLStorageCheck [386] |
| Description | Confirms the THL storage directory exists, is empty and writeable |

Confirms that the directory configured for THL storage using `--log-dir` directory exists, is writeable, and is empty.

– THLStorageChecksum [386]

| | |
|---|---|
| Option | THLStorageChecksum [386] |
| Description | |

– TargetDirectoryDoesNotExist [386]

| | |
|---|---|
| Option | TargetDirectoryDoesNotExist [386] |
| Description | |

– TransferredLogStorageCheck [386]

| | |
|---|---|
| Option | TransferredLogStorageCheck [386] |
| Description | |

– UpgradeSameProductCheck [386]

| | |
|---|---|
| Option | UpgradeSameProductCheck [386] |
| Description | Ensures that the same product is being updated |

Updates must occur with the same product, for example, Tungsten Replicator to Tungsten Replicator. It is not possible to update replicator to cluster, or cluster to replicator.

– VIPEnabledHostAllowsRootCommands [386]

| | |
|---|---|
| Option | VIPEnabledHostAllowsRootCommands [386] |
| Description | |

– VIPEnabledHostArpPath [386]

| | |
|---|---|
| Option | VIPEnabledHostArpPath [386] |
| Description | |

– VIPEnabledHostIfconfigPath [386]

| Option | VIPEnabledHostIfconfigPath [386] |
|---|---|
| Description | |

– VerticaUserGroupsCheck [387]

| Option | VerticaUserGroupsCheck [387] |
|---|---|
| Description | Checks that the Vertica user has the correct OS group membership |

Checks whether the user running Vertica is a member of the tungsten user's primary group. Without this setting, the CSV files generated by the replicator would not be readable by Vertica when importing them into the database during batchloading.

– WhichAvailableCheck [387]

| Option | WhichAvailableCheck [387] |
|---|---|
| Description | Checks the existence of a working which command |

Checks the existence of a working which command.

– WriteableHomeDirectoryCheck [387]

| Option | WriteableHomeDirectoryCheck [387] |
|---|---|
| Description | Ensures the home directory can be written to |

Checks that the home directory for the configured user can be written to.

– WriteableTempDirectoryCheck [387]

| Option | WriteableTempDirectoryCheck [387] |
|---|---|
| Description | Ensures the temporary directory can be written to |

The temporary directory is used during installation to store a variety of information. This check ensures that the directory is writeable, and that files can be created and deleted correctly.

– XtrabackupAvailableCheck [387]

| Option | XtrabackupAvailableCheck [387] |
|---|---|
| Description | |

– XtrabackupDirectoryWriteableCheck [387]

| Option | XtrabackupDirectoryWriteableCheck [387] |
|---|---|
| Description | |

– XtrabackupSettingsCheck [387]

| Option | XtrabackupSettingsCheck [387] |
|---|---|
| Description | |

# 9.8. tpm Configuration Options

tpm supports a large range of configuration options, which can be specified either:

- On the command-line, using a double-dash prefix, i.e. --repl-thl-log-retention=3d [428]
- In an INI file, without the double-dash prefix, i.e. repl-thl-log-retention=3d [428]

A full list of all the available options supported is provided in Table 9.20, “tpm Configuration Options”.

Table 9.20. tpm Configuration Options

| CmdLine Option | INI File Option | Description |
|---|---|---|
| --auto-enable [395], --repl-auto-enable [395] | auto-enable [395], repl-auto-enable [395] | Auto-enable services after start-up |
| --auto-recovery-delay-interval [395], --repl-auto-recovery-delay-interval [395] | auto-recovery-delay-interval [395], repl-auto-recovery-delay-interval [395] | Delay (in seconds) between going OFFLINE and attempting to go ONLINE |
| --auto-recovery-max-attempts [395], --repl-auto-recovery-max-attempts [395] | auto-recovery-max-attempts [395], repl-auto-recovery-max-attempts [395] | Maximum number of attempts at automatic recovery |
| --auto-recovery-reset-interval [396], --repl-auto-recovery-reset-interval [396] | auto-recovery-reset-interval [396], repl-auto-recovery-reset-interval [396] | Delay (in seconds) before autorecovery is deemed to have succeeded |
| --backup-directory [396], --repl-backup-directory [396] | backup-directory [396], repl-backup-directory [396] | Permanent backup storage directory |
| --backup-dump-directory [396], --repl-backup-dump-directory [396] | backup-dump-directory [396], repl-backup-dump-directory [396] | Backup temporary dump directory |
| --backup-method [396], --repl-backup-method [396] | backup-method [396], repl-backup-method [396] | Database backup method |
| --backup-online [397], --repl-backup-online [397] | backup-online [397], repl-backup-online [397] | Does the backup script support backing up a datasource while it is ONLINE |
| --backup-options [397] | backup-options [397] | Space separated list of options to pass directly to the underlying backup binary in use. |
| --backup-retention [397], --repl-backup-retention [397] | backup-retention [397], repl-backup-retention [397] | Number of backups to retain |
| --backup-script [397], --repl-backup-script [397] | backup-script [397], repl-backup-script [397] | What is the path to the backup script |
| --batch-enabled [397] | batch-enabled [397] | Should the replicator service use a batch applier |
| --batch-load-language [398] | batch-load-language [398] | Which script language to use for batch loading |
| --batch-load-template [398] | batch-load-template [398] | Value for the loadBatchTemplate property |
| --repl-buffer-size [398] | repl-buffer-size [398] | Replicator queue size between stages (min 1) |
| --channels [398], --repl-channels [398] | channels [398], repl-channels [398] | Number of replication channels to use for parallel apply. |
| --cluster-slave-auto-recovery-delay-interval [398], --cluster-slave-repl-auto-recovery-delay-interval [398] | cluster-slave-auto-recovery-delay-interval [398], cluster-slave-repl-auto-recovery-delay-interval [398] | Default value for --auto-recovery-delay-interval when --topology=cluster-slave |
| --cluster-slave-auto-recovery-max-attempts [398], --cluster-slave-repl-auto-recovery-max-attempts [398] | cluster-slave-auto-recovery-max-attempts [398], cluster-slave-repl-auto-recovery-max-attempts [398] | Default value for --auto-recovery-max-attempts when --topology=cluster-slave |
| --cluster-slave-auto-recovery-reset-interval [398], --cluster-slave-repl-auto-recovery-reset-interval [398] | cluster-slave-auto-recovery-reset-interval [398], cluster-slave-repl-auto-recovery-reset-interval [398] | Default value for --auto-recovery-reset-interval when --topology=cluster-slave |
| --config-file [399] | config-file [399] | Display help information for content of the config file |
| --consistency-policy [399], --repl-consistency-policy [399] | consistency-policy [399], repl-consistency-policy [399] | Should the replicator stop or warn if a consistency check fails? |
| --dataservice-name [399] | dataservice-name [399] | Limit the command to the hosts in this dataservice Multiple data services may be specified by providing a comma separated list |
| --dataservice-relay-enabled [399] | dataservice-relay-enabled [399] | Make this dataservice a replica of another |
| --dataservice-schema [399] | dataservice-schema [399] | The db schema to hold dataservice details |
| --dataservice-thl-port [399] | dataservice-thl-port [399] | Port to use for THL operations |
| --dataservice-vip-enabled [399] | dataservice-vip-enabled [399] | Is VIP management enabled? |

| CmdLine Option | INI File Option | Description |
|---|---|---|
| --dataservice-vip-ipaddress [400] | dataservice-vip-ipaddress [400] | VIP IP address |
| --dataservice-vip-netmask [400] | dataservice-vip-netmask [400] | VIP netmask |
| --datasource-boot-script [400], --repl-datasource-boot-script [400] | datasource-boot-script [400], repl-datasource-boot-script [400] | Database start script |
| --datasource-enable-ssl [400], --repl-datasource-enable-ssl [400] | datasource-enable-ssl [400], repl-datasource-enable-ssl [400] | Enable SSL connection to DBMS server |
| --datasource-group-id [400] | datasource-group-id [400] | All nodes within a cluster that share the same datasource-group-id will form a Distributed Datasource Group (DDG). |
| --datasource-log-directory [400], --repl-datasource-log-directory [400] | datasource-log-directory [400], repl-datasource-log-directory [400] | Primary log directory |
| --datasource-log-pattern [400], --repl-datasource-log-pattern [400] | datasource-log-pattern [400], repl-datasource-log-pattern [400] | Primary log filename pattern |
| --datasource-mysql-conf [401], --repl-datasource-mysql-conf [401] | datasource-mysql-conf [401], repl-datasource-mysql-conf [401] | MySQL config file |
| --datasource-mysql-data-directory [401], --repl-datasource-mysql-data-directory [401] | datasource-mysql-data-directory [401], repl-datasource-mysql-data-directory [401] | MySQL data directory |
| --datasource-mysql-ibdata-directory [401], --repl-datasource-mysql-ibdata-directory [401] | datasource-mysql-ibdata-directory [401], repl-datasource-mysql-ibdata-directory [401] | MySQL InnoDB data directory |
| --datasource-mysql-iblog-directory [401], --repl-datasource-mysql-iblog-directory [401] | datasource-mysql-iblog-directory [401], repl-datasource-mysql-iblog-directory [401] | MySQL InnoDB log directory |
| --datasource-mysql-ssl-ca [401], --repl-datasource-mysql-ssl-ca [401] | datasource-mysql-ssl-ca [401], repl-datasource-mysql-ssl-ca [401] | MySQL SSL CA file |
| --datasource-mysql-ssl-cert [401], --repl-datasource-mysql-ssl-cert [401] | datasource-mysql-ssl-cert [401], repl-datasource-mysql-ssl-cert [401] | MySQL SSL certificate file |
| --datasource-mysql-ssl-key [401], --repl-datasource-mysql-ssl-key [401] | datasource-mysql-ssl-key [401], repl-datasource-mysql-ssl-key [401] | MySQL SSL key file |
| --datasource-oracle-service [402], --repl-datasource-oracle-service [402] | datasource-oracle-service [402], repl-datasource-oracle-service [402] | Oracle Service Name |
| --datasource-systemctl-service [402], --repl-datasource-systemctl-service [402] | datasource-systemctl-service [402], repl-datasource-systemctl-service [402] | Database systemctl script |
| --datasource-type [402], --repl-datasource-type [402] | datasource-type [402], repl-datasource-type [402] | Database type |
| --delete [402] | delete [402] | Delete the named data service from the configuration Data Service options |
| --deploy-systemd [402] | deploy-systemd [402] | Setting to true will deploy and configure software to be controlled by systemd. |
| --direct-datasource-log-directory [403], --repl-direct-datasource-log-directory [403] | direct-datasource-log-directory [403], repl-direct-datasource-log-directory [403] | Primary log directory |
| --direct-datasource-log-pattern [403], --repl-direct-datasource-log-pattern [403] | direct-datasource-log-pattern [403], repl-direct-datasource-log-pattern [403] | Primary log filename pattern |
| --direct-datasource-type [403], --repl-direct-datasource-type [403] | direct-datasource-type [403], repl-direct-datasource-type [403] | Database type |
| --direct-replication-host [403], --direct-datasource-host [403], --repl-direct-datasource-host [403] | direct-datasource-host [403], direct-replication-host [403], repl-direct-datasource-host [403] | Database server hostname |

| CmdLine Option | INI File Option | Description |
|---|---|---|
| --direct-replication-password [403], --direct-datasource-password [403], --repl-direct-datasource-password [403] | direct-datasource-password [403], direct-replication-password [403], repl-direct-datasource-password [403] | Password for datasource connection |
| --direct-replication-port [403], --direct-datasource-port [403], --repl-direct-datasource-port [403] | direct-datasource-port [403], direct-replication-port [403], repl-direct-datasource-port [403] | Database server port |
| --direct-replication-user [404], --direct-datasource-user [404], --repl-direct-datasource-user [404] | direct-datasource-user [404], direct-replication-user [404], repl-direct-datasource-user [404] | Database login for Tungsten |
| --directory [404] | directory [404] | Set the directory of an existing installation used during fetching an existing configuration |
| --disable-relay-logs [404], --repl-disable-relay-logs [404] | disable-relay-logs [404], repl-disable-relay-logs [404] | Disable the use of relay-logs? |
| --disable-security-controls [404] | disable-security-controls [404] | Disables all forms of security, including SSL, TLS and authentication |
| --disable-slave-extractor [404], --repl-disable-slave-extractor [404] | disable-slave-extractor [404], repl-disable-slave-extractor [404] | Should replica servers support the primary role? |
| --drop-static-columns-in-updates [404] | drop-static-columns-in-updates [404] | This will modify UPDATE transactions in row-based replication and eliminate any columns that were not modified. |
| --enable-active-witnesses [405], --active-witnesses [405] | active-witnesses [405], enable-active-witnesses [405] | Enable active witness hosts |
| --enable-batch-master [405] | enable-batch-master [405] | Enable batch operation for the primary |
| --enable-batch-service [405] | enable-batch-service [405] | Enables batch mode for a service |
| --enable-batch-slave [406] | enable-batch-slave [406] | Enable batch operation for the Replica |
| --enable-heterogeneous-master [406] | enable-heterogeneous-master [406] | Enable heterogeneous operation for the primary |
| --enable-heterogeneous-service [406] | enable-heterogeneous-service [406] | Enable heterogeneous operation |
| --enable-heterogeneous-slave [406] | enable-heterogeneous-slave [406] | Enable heterogeneous operation for the replica |
| --enable-jgroups-ssl [407], --jgroups-ssl [407] | enable-jgroups-ssl [407], jgroups-ssl [407] | Enable SSL encryption of JGroups communication on this host |
| --enable-rmi-authentication [407], --rmi-authentication [407] | enable-rmi-authentication [407], rmi-authentication [407] | Enable RMI authentication for the services running on this host |
| --enable-rmi-ssl [407], --rmi-ssl [407] | enable-rmi-ssl [407], rmi-ssl [407] | Enable SSL encryption of RMI communication on this host |
| --enable-slave-thl-listener [407], --repl-enable-slave-thl-listener [407] | enable-slave-thl-listener [407], repl-enable-slave-thl-listener [407] | Should this service allow THL connections? |
| --enable-sudo-access [407], --root-command-prefix [407] | enable-sudo-access [407], root-command-prefix [407] | Run root commands using sudo |
| --enable-thl-ssl [408], --repl-enable-thl-ssl [408], --thl-ssl [408] | enable-thl-ssl [408], repl-enable-thl-ssl [408], thl-ssl [408] | Enable SSL encryption of THL communication for this service |
| --executable-prefix [408] | executable-prefix [408] | Adds a prefix to command aliases |
| --file-protection-level [408] | file-protection-level [408] | Protection level for Continuent files |
| --file-protection-umask [408] | file-protection-umask [408] | Protection umask for Continuent files |
| --host-name [408] | host-name [408] | DNS hostname |
| --hosts [409] | hosts [409] | Limit the command to the hosts listed You must use the hostname as it appears in the configuration. |
| --hub [409], --dataservice-hub-host [409] | dataservice-hub-host [409], hub [409] | What is the hub host for this all-masters dataservice? |

| CmdLine Option | INI File Option | Description |
|---|---|---|
| --hub-service [409], --dataservice-hub-service [409] | dataservice-hub-service [409], hub-service [409] | The data service to use for the hub of a star topology |
| --install [409] | install [409] | Install service start scripts |
| --install-directory [409], --home-directory [409] | home-directory [409], install-directory [409] | Installation directory |
| --java-enable-concurrent-gc [409], --repl-java-enable-concurrent-gc [409] | java-enable-concurrent-gc [409], repl-java-enable-concurrent-gc [409] | Replicator Java uses concurrent garbage collection |
| --java-external-lib-dir [409], --repl-java-external-lib-dir [409] | java-external-lib-dir [409], repl-java-external-lib-dir [409] | Directory for 3rd party Jar files required by replicator |
| --java-file-encoding [410], --repl-java-file-encoding [410] | java-file-encoding [410], repl-java-file-encoding [410] | Java platform charset (esp. for heterogeneous replication) |
| --java-jgroups-key [410] | java-jgroups-key [410] | The alias to use for the JGroups TLS key in the keystore. |
| --java-jgroups-keystore-path [410] | java-jgroups-keystore-path [410] | Local path to the JGroups Java Keystore file. |
| --java-jmxremote-access-path [410] | java-jmxremote-access-path [410] | Local path to the Java JMX Remote Access file. |
| --java-keystore-password [410] | java-keystore-password [410] | Set the password for unlocking the tungsten_keystore.jks file in the security directory. Specific for intra cluster communication. |
| --java-keystore-path [410] | java-keystore-path [410] | Local path to the Java Keystore file. Specific for intra cluster communication. NOTE: When java-keystore-path is passed to tpm, the keystore must contain both tls and mysql certs when appropriate. tpm will NOT add mysql cert nor generate tls cert when this flag is found, so both certs must be manually imported already. |
| --java-mem-size [410], --repl-java-mem-size [410] | java-mem-size [410], repl-java-mem-size [410] | Replicator Java heap memory size in Mb (min 128) |
| --java-passwordstore-path [411] | java-passwordstore-path [411] | Local path to the Java Password Store file. |
| --java-tls-alias [411] | java-tls-alias [411] | The alias to use for the TLS key/certificate in the keystore and truststore. |
| --java-tls-key-lifetime [411] | java-tls-key-lifetime [411] | Lifetime for the Java TLS key |
| --java-tls-keystore-path [411] | java-tls-keystore-path [411] | The keystore holding a certificate to use for all Continuent TLS encryption. |
| --java-truststore-password [411] | java-truststore-password [411] | The password for unlocking the tungsten_truststore.jks file in the security directory |
| --java-truststore-path [411] | java-truststore-path [411] | Local path to the Java Truststore file. |
| --java-user-timezone [411], --repl-java-user-timezone [411] | java-user-timezone [411], repl-java-user-timezone [411] | Java VM Timezone (esp. for cross-site replication) |
| --log [412] | log [412] | Write all messages, visible and hidden, to this file. You may specify a filename, 'pid' or 'timestamp'. |
| --log-slave-updates [412] | log-slave-updates [412] | Should replicas log updates to binlog |
| --master [412], --dataservice-master-host [412], --masters [412], --relay [412] | dataservice-master-host [412], master [412], masters [412], relay [412] | Hostname of the primary (or relay) host within this service |
| --master-preferred-role [412], --repl-master-preferred-role [412] | master-preferred-role [412], repl-master-preferred-role [412] | Preferred role for primary THL when connecting as a replica |
| --master-services [412], --dataservice-master-services [412] | dataservice-master-services [412], master-services [412] | Data service names that should be used on each Primary |
| --master-thl-host [412] | master-thl-host [412] | Primary THL Hostname |
| --master-thl-port [413] | master-thl-port [413] | Primary THL Port |
| --members [413], --dataservice-hosts [413] | dataservice-hosts [413], members [413] | Hostnames for the dataservice members |
| --metadata-directory [413], --repl-metadata-directory [413] | metadata-directory [413], repl-metadata-directory [413] | Replicator metadata directory |

| CmdLine Option | INI File Option | Description |
|---|---|---|
| --mysql-allow-intensive-checks [413] | mysql-allow-intensive-checks [413] | For MySQL installation, enables detailed checks on the supported data types within the MySQL database to confirm compatibility. |
| --mysql-driver [413] | mysql-driver [413] | MySQL Driver Vendor |
| --mysql-enable-ansiquotes [413], --repl-mysql-enable-ansiquotes [413] | mysql-enable-ansiquotes [413], repl-mysql-enable-ansiquotes [413] | Enables ANSI_QUOTES mode for incoming events? |
| --mysql-enable-enumtostring [414], --repl-mysql-enable-enumtostring [414] | mysql-enable-enumtostring [414], repl-mysql-enable-enumtostring [414] | Enable a filter to convert ENUM values to strings |
| --mysql-enable-noonlykeywords [414], --repl-mysql-enable-noonlykeywords [414] | mysql-enable-noonlykeywords [414], repl-mysql-enable-noonlykeywords [414] | Enables a filter to translate DELETE FROM ONLY to DELETE FROM and UPDATE ONLY to UPDATE. |
| --mysql-enable-settostring [414], --repl-mysql-enable-settostring [414] | mysql-enable-settostring [414], repl-mysql-enable-settostring [414] | Enable a filter to convert SET types to strings |
| --mysql-ro-slave [414], --repl-mysql-ro-slave [414] | mysql-ro-slave [414], repl-mysql-ro-slave [414] | Replicas are read-only? |
| --mysql-server-id [414], --repl-mysql-server-id [414] | mysql-server-id [414], repl-mysql-server-id [414] | Explicitly set the MySQL server ID |
| --mysql-use-bytes-for-string [414], --repl-mysql-use-bytes-for-string [414] | mysql-use-bytes-for-string [414], repl-mysql-use-bytes-for-string [414] | Transfer strings as their byte representation? |
| --mysql-xtrabackup-dir [414], --repl-mysql-xtrabackup-dir [414] | mysql-xtrabackup-dir [414], repl-mysql-xtrabackup-dir [414] | Directory to use for storing xtrabackup full & incremental backups |
| --native-slave-takeover [415], --repl-native-slave-takeover [415] | native-slave-takeover [415], repl-native-slave-takeover [415] | Takeover native replication |
| --no-connectors [415] | no-connectors [415] | When issued during an update, connectors will not be restarted. Restart of the connectors will then need to be performed mannually for updates to take affect. |
| --no-deployment [415] | no-deployment [415] | Skip deployment steps that create the install directory |
| --no-validation [415] | no-validation [415] | Skip validation checks that run on each host |
| --optimize-row-events [415] | optimize-row-events [415] | Enables or disables optimized row updates. Enabled by default. |
| --optimize-row-events-limit-delete-rows [415] | optimize-row-events-limit-delete-rows [415] | Limits the number of deletes grouped per event when optimize-row-events is enabled. Note that deletes can only be optimized for tables with single column PK's |
| --optimize-row-events-limit-insert-rows [416] | optimize-row-events-limit-insert-rows [416] | Limits the number of inserts grouped per event when optimize-row-events is enabled. |
| --postgresql-dbname [416], --repl-postgresql-dbname [416] | postgresql-dbname [416], repl-postgresql-dbname [416] | Name of the database to replicate |
| --prefer-ip-stack [416] | prefer-ip-stack [416] | Switch between IPv4 and IPv6 support. |
| --preferred-path [416] | preferred-path [416] | Additional command path |
| --prefetch-enabled [416] | prefetch-enabled [416] | Should the replicator service be setup as a prefetch applier |
| --prefetch-max-time-ahead [417] | prefetch-max-time-ahead [417] | Maximum number of seconds that the prefetch applier can get in front of the standard applier |
| --prefetch-min-time-ahead [417] | prefetch-min-time-ahead [417] | Minimum number of seconds that the prefetch applier must be in front of the standard applier |
| --prefetch-schema [417] | prefetch-schema [417] | Schema to watch for timing prefetch progress |
| --prefetch-sleep-time [417] | prefetch-sleep-time [417] | How long to wait when the prefetch applier gets too far ahead |
| --privileged-master [417] | privileged-master [417] | Does the login for the Primary database service have superuser privileges |
| --privileged-slave [417] | privileged-slave [417] | Does the login for the Replica database service have superuser privileges |
| --profile-script [417] | profile-script [417] | Append commands to include env.sh in this profile script |

| CmdLine Option | INI File Option | Description |
|---|---|---|
| --protect-configuration-files [417] | protect-configuration-files [417] | When enabled, configuration files are protected to be only readable and updatable by the configured user |
| --redshift-dbname [418], --repl-redshift-dbname [418] | redshift-dbname [418], repl-redshift-dbname [418] | Name of the Redshift database to replicate into |
| --relay-directory [418], --repl-relay-directory [418] | relay-directory [418], repl-relay-directory [418] | Directory for logs transferred from the Primary |
| --relay-enabled [418] | relay-enabled [418] | Should the replicator service be setup as a relay. |
| --relay-source [419], --dataservice-relay-source [419], --master-dataservice [419] | dataservice-relay-source [419], master-dataservice [419], relay-source [419] | Dataservice name to use as a relay source |
| --repl-allow-bidi-unsafe [419] | repl-allow-bidi-unsafe [419] | Allow unsafe SQL from remote service |
| --repl-store-thl-compressed [419] | repl-store-thl-compressed [419] | Enable (true) or disable (false) compression of THL on disk. |
| --repl-store-thl-encrypted [419] | repl-store-thl-encrypted [419] | Enable (true) or disable (false) encryption of THL on disk. |
| --repl-svc-extractor-multi-frag-service-detection [419] | repl-svc-extractor-multi-frag-service-detection [419] | Force extraction to read ahead to last fragment to detect service name |
| --repl-thl-client-serialization [420] | repl-thl-client-serialization [420] | Enable THL compression on downstream Replicator Appliers. |
| --repl-thl-server-serialization [420] | repl-thl-server-serialization [420] | Comma Separated list of THL compression protocols to enable on the thl-server (Extractor). |
| --replace-tls-certificate [420] | replace-tls-certificate [420] | Replace the TLS certificate |
| --replication-host [420], --datasource-host [420], --repl-datasource-host [420] | datasource-host [420], repl-datasource-host [420], replication-host [420] | Hostname of the datasource |
| --replication-password [420], --datasource-password [420], --repl-datasource-password [420] | datasource-password [420], repl-datasource-password [420], replication-password [420] | Database password |
| --replication-port [421], --datasource-port [421], --repl-datasource-port [421] | datasource-port [421], repl-datasource-port [421], replication-port [421] | Database network port |
| --replication-user [421], --datasource-user [421], --repl-datasource-user [421] | datasource-user [421], repl-datasource-user [421], replication-user [421] | User for database connection |
| --replicator-rest-api [421] | replicator-rest-api [421] | Enable (default) or Disable APIv2 |
| --replicator-rest-api-address [421] | replicator-rest-api-address [421] | Address for the API to bind too. |
| --replicator-rest-api-authentication [421] | replicator-rest-api-authentication [421] | Enforce authentication for the API. |
| --replicator-rest-api-ssl [421] | replicator-rest-api-ssl [421] | Enable SSL for the API. |
| --reset [422] | reset [422] | Clear the current configuration before processing any arguments |
| --rest-api-admin-pass [422], --rest-api-admin-password [422] | rest-api-admin-pass [422], rest-api-admin-password [422] | Specify the initial Admin User Password for API access. rest-api-admin-password alias only available from version 7.1.2 onwards. |
| --rest-api-admin-user [422] | rest-api-admin-user [422] | Specify the initial Admin Username for API access |
| --rmi-port [422], --repl-rmi-port [422] | repl-rmi-port [422], rmi-port [422] | Replication RMI listen port |
| --rmi-user [422] | rmi-user [422] | The username for RMI authentication |
| --role [422], --repl-role [422] | repl-role [422], role [422] | What is the replication role for this service? |
| --security-directory [423] | security-directory [423] | Storage directory for the Java security/encryption files |
| --service-alias [423], --dataservice-service-alias [423] | dataservice-service-alias [423], service-alias [423] | Replication alias of this dataservice |
| --service-name [423] | service-name [423] | Set the service name |

| CmdLine Option | INI File Option | Description |
|---|---|---|
| --service-type [423], --repl-service-type [423] | repl-service-type [423], service-type [423] | What is the replication service type? |
| --skip-statemap [423] | skip-statemap [423] | Do not copy the cluster-home/conf/statemap.properties from the previous install |
| --slaves [423], --dataservice-slaves [423], --members [413] | dataservice-slaves [423], members [413], slaves [423] | What are the Replicas for this dataservice? |
| --start [424] | start [424] | Start the services after configuration |
| --start-and-report [424] | start-and-report [424] | Start the services and report out the status after configuration |
| --svc-allow-any-remote-service [424], --repl-svc-allow-any-remote-service [424] | repl-svc-allow-any-remote-service [424], svc-allow-any-remote-service [424] | Replicate from any service |
| --svc-applier-block-commit-interval [424], --repl-svc-applier-block-commit-interval [424] | repl-svc-applier-block-commit-interval [424], svc-applier-block-commit-interval [424] | Minimum interval between commits |
| --svc-applier-block-commit-size [424], --repl-svc-applier-block-commit-size [424] | repl-svc-applier-block-commit-size [424], svc-applier-block-commit-size [424] | Applier block commit size (min 1) |
| --svc-applier-filters [424], --repl-svc-applier-filters [424] | repl-svc-applier-filters [424], svc-applier-filters [424] | Replication service applier filters |
| --svc-applier-last-applied-write-interval [425] | svc-applier-last-applied-write-interval [425] | Interval (in seconds) to store the last known applied seqno and latency |
| --svc-extractor-filters [425], --repl-svc-extractor-filters [425] | repl-svc-extractor-filters [425], svc-extractor-filters [425] | Replication service extractor filters |
| --svc-fail-on-zero-row-update [425], --repl-svc-fail-on-zero-row-update [425] | repl-svc-fail-on-zero-row-update [425], svc-fail-on-zero-row-update [425] | How should the replicator behave when a Row-Based Replication UPDATE or DELETE does not affect any rows. |
| --svc-parallelization-type [425], --repl-svc-parallelization-type [425] | repl-svc-parallelization-type [425], svc-parallelization-type [425] | Method for implementing parallel apply |
| --svc-remote-filters [425], --repl-svc-remote-filters [425] | repl-svc-remote-filters [425], svc-remote-filters [425] | Replication service remote download filters |
| --svc-reposition-on-source-id-change [426], --repl-svc-reposition-on-source-id-change [426] | repl-svc-reposition-on-source-id-change [426], svc-reposition-on-source-id-change [426] | The Primary will come ONLINE from the current position if the stored source_id does not match the value in the static properties |
| --svc-shard-default-db [426], --repl-svc-shard-default-db [426] | repl-svc-shard-default-db [426], svc-shard-default-db [426] | Mode for setting the shard ID from the default db |
| --svc-systemd-config-replicator [426] | svc-systemd-config-replicator [426] | Used to provide custom systemd configuration that will be used in place of the default generated on. |
| --svc-table-engine [426], --repl-svc-table-engine [426] | repl-svc-table-engine [426], svc-table-engine [426] | Replication service table engine |
| --svc-thl-filters [426], --repl-svc-thl-filters [426] | repl-svc-thl-filters [426], svc-thl-filters [426] | Replication service THL filters |
| --target-dataservice [427], --slave-dataservice [427] | slave-dataservice [427], target-dataservice [427] | Dataservice to use to determine the value of host configuration |
| --temp-directory [427] | temp-directory [427] | Temporary Directory |
| --template-file [427] | template-file [427] | Display the keys that may be used in configuration template files |
| --template-search-path [427] | template-search-path [427] | Adds a new template search path for configuration file generation |
| --thl-directory [427], --repl-thl-directory [427] | repl-thl-directory [427], thl-directory [427] | Replicator log directory |
| --thl-do-checksum [428], --repl-thl-do-checksum [428] | repl-thl-do-checksum [428], thl-do-checksum [428] | Execute checksum operations on THL log files |

| CmdLine Option | INI File Option | Description |
|---|---|---|
| --thl-interface [428], --repl-thl-interface [428] | repl-thl-interface [428], thl-interface [428] | Listen interface to use for THL operations |
| --thl-log-connection-timeout [428], --repl-thl-log-connection-timeout [428] | repl-thl-log-connection-timeout [428], thl-log-connection-timeout [428] | Number of seconds to wait for a connection to the THL log |
| --thl-log-file-size [428], --repl-thl-log-file-size [428] | repl-thl-log-file-size [428], thl-log-file-size [428] | File size in bytes for THL disk logs |
| --thl-log-fsync [428], --repl-thl-log-fsync [428] | repl-thl-log-fsync [428], thl-log-fsync [428] | Fsync THL records on commit. More reliable operation but adds latency to replication when using low-performance storage |
| --thl-log-retention [428], --repl-thl-log-retention [428] | repl-thl-log-retention [428], thl-log-retention [428] | How long do you want to keep THL files. |
| --thl-port [429], --repl-thl-port [429] | repl-thl-port [429], thl-port [429] | Port to use for THL Operations |
| --thl-protocol [429], --repl-thl-protocol [429] | repl-thl-protocol [429], thl-protocol [429] | Protocol to use for THL communication with this service |
| --topology [429], --dataservice-topology [429] | dataservice-topology [429], topology [429] | Replication topology for the dataservice. |
| --track-schema-changes [429] | track-schema-changes [429] | This will enable filters that track DDL statements and write the resulting change to files on Replica hosts. The feature is intended for use in some batch deployments. |
| --user [429] | user [429] | System User |
| --vertica-dbname [430], --repl-vertica-dbname [430] | repl-vertica-dbname [430], vertica-dbname [430] | Name of the database to replicate into |
| --witnesses [430], --dataservice-witnesses [430] | dataservice-witnesses [430], witnesses [430] | Witness hosts for the dataservice |

## 9.8.1. A tpm Options

--auto-enable

| | | |
|---|---|---|
| Option | --auto-enable [395] | |
| Aliases | --repl-auto-enable [395] | |
| Config File Options | auto-enable [395], repl-auto-enable [395] | |
| Description | Auto-enable services after start-up | |
| Value Type | boolean | |
| Valid Values | false | |
| | true | |

--auto-recovery-delay-interval

| | |
|---|---|
| Option | --auto-recovery-delay-interval [395] |
| Aliases | --repl-auto-recovery-delay-interval [395] |
| Config File Options | auto-recovery-delay-interval [395], repl-auto-recovery-delay-interval [395] |
| Description | Delay (in seconds) between going OFFLINE and attempting to go ONLINE |
| Value Type | numeric |
| Default | 5 |

The delay between the replicator identifying that autorecovery is needed, and autorecovery being attempted. For busy MySQL installations, larger numbers may be needed to allow time for MySQL servers to restart or recover from their failure.

--auto-recovery-max-attempts

| | |
|---|---|
| Option | --auto-recovery-max-attempts [395] |
| Aliases | --repl-auto-recovery-max-attempts [395] |

| | |
|---|---|
| Config File Options | auto-recovery-max-attempts [395], repl-auto-recovery-max-attempts [395] |
| Description | Maximum number of attempts at automatic recovery |
| Value Type | numeric |
| Default | 0 |

Specifies the number of attempts the replicator will make to go back online. When the number of attempts has been reached, the replicator will remain in the OFFLINE [195] state.

Autorecovery is not enabled until the value of this parameter is set to a non-zero value. The state of autorecovery can be determined using the *autoRecoveryEnabled* status parameter. The number of attempts made to autorecover can be tracked using the *autoRecoveryTotal* status parameter.

--auto-recovery-reset-interval

| | |
|---|---|
| Option | --auto-recovery-reset-interval [396] |
| Aliases | --repl-auto-recovery-reset-interval [396] |
| Config File Options | auto-recovery-reset-interval [396], repl-auto-recovery-reset-interval [396] |
| Description | Delay (in seconds) before autorecovery is deemed to have succeeded |
| Value Type | numeric |
| Default | 5 |

The time in ONLINE [195] state that indicates to the replicator that the autorecovery procedure has succeeded. For servers with very large transactions, this value should be increased to allow the transaction to be successfully applied.

## 9.8.2. B tpm Options

--backup-directory

| | |
|---|---|
| Option | --backup-directory [396] |
| Aliases | --repl-backup-directory [396] |
| Config File Options | backup-directory [396], repl-backup-directory [396] |
| Description | Permanent backup storage directory |
| Value Type | string |
| Default | {home directory}/backups |

--backup-dump-directory

| | |
|---|---|
| Option | --backup-dump-directory [396] |
| Aliases | --repl-backup-dump-directory [396] |
| Config File Options | backup-dump-directory [396], repl-backup-dump-directory [396] |
| Description | Backup temporary dump directory |
| Value Type | string |

--backup-method

| | | |
|---|---|---|
| Option | --backup-method [396] | |
| Aliases | --repl-backup-method [396] | |
| Config File Options | backup-method [396], repl-backup-method [396] | |
| Description | Database backup method | |
| Value Type | string | |
| Valid Values | ebs-snapshot | |
| | file-copy-snapshot | |
| | mariabackup | Use mariabackup (Available from v7.0.0 only) |
| | mariabackup-incremental | Use mariabackup (Available from v7.0.0 only) |

| | | |
|---|---|---|
| | mysqldump | Use mysqldump |
| | none | |
| | script | Use a custom script |
| | xtrabackup | Use Percona XtraBackup |
| | xtrabackup-full | Use Percona XtraBackup Full |
| | xtrabackup-incremental | Use Percona XtraBackup Incremental |

The default, if not supplied, will be dependant on the enviroment. During installation tpm will detect which tools are available, favouring xtrabackup-full (or mariabackup-full). If not found, then mysqldump will be the default.

--backup-online

| | | |
|---|---|---|
| Option | `--backup-online` [397] | |
| Aliases | `--repl-backup-online` [397] | |
| Config File Options | `backup-online` [397], `repl-backup-online` [397] | |
| Description | Does the backup script support backing up a datasource while it is ONLINE | |
| Value Type | boolean | |
| Valid Values | false | |
| | true | |

--backup-options

| | |
|---|---|
| Option | `--backup-options` [397] |
| Config File Options | `backup-options` [397] |
| Description | Space separated list of options to pass directly to the underlying backup binary in use. |
| Value Type | string |

Options passed are not validated by Tungsten. They are specific to the backup binary you choose to use and therefore you must ensure accuracy in syntax.

--backup-retention

| | |
|---|---|
| Option | `--backup-retention` [397] |
| Aliases | `--repl-backup-retention` [397] |
| Config File Options | `backup-retention` [397], `repl-backup-retention` [397] |
| Description | Number of backups to retain |
| Value Type | numeric |

--backup-script

| | |
|---|---|
| Option | `--backup-script` [397] |
| Aliases | `--repl-backup-script` [397] |
| Config File Options | `backup-script` [397], `repl-backup-script` [397] |
| Description | What is the path to the backup script |
| Value Type | filename |

--batch-enabled

| | | |
|---|---|---|
| Option | `--batch-enabled` [397] | |
| Config File Options | `batch-enabled` [397] | |
| Description | Should the replicator service use a batch applier | |
| Value Type | boolean | |
| Default | false | |
| Valid Values | true | |

--batch-load-language

| | | |
|---|---|---|
| Option | `--batch-load-language` [398] | |
| Config File Options | `batch-load-language` [398] | |
| Description | Which script language to use for batch loading | |
| Value Type | string | |
| Valid Values | js | JavaScript |
| | sql | SQL |

--batch-load-template

| | |
|---|---|
| Option | `--batch-load-template` [398] |
| Config File Options | `batch-load-template` [398] |
| Description | Value for the loadBatchTemplate property |
| Value Type | string |

--repl-buffer-size

| | |
|---|---|
| Option | `--repl-buffer-size` [398] |
| Config File Options | `repl-buffer-size` [398] |
| Description | Replicator queue size between stages (min 1) |
| Value Type | numeric |
| Default | 10 |

## 9.8.3. C tpm Options

--channels

| | |
|---|---|
| Option | `--channels` [398] |
| Aliases | `--repl-channels` [398] |
| Config File Options | `channels` [398], `repl-channels` [398] |
| Description | Number of replication channels to use for parallel apply. |
| Value Type | numeric |
| Default | 1 |

--cluster-slave-auto-recovery-delay-interval

| | |
|---|---|
| Option | `--cluster-slave-auto-recovery-delay-interval` [398] |
| Aliases | `--cluster-slave-repl-auto-recovery-delay-interval` [398] |
| Config File Options | `cluster-slave-auto-recovery-delay-interval` [398], `cluster-slave-repl-auto-recovery-delay-interval` [398] |
| Description | Default value for --auto-recovery-delay-interval when --topology=cluster-slave |
| Value Type | string |

--cluster-slave-auto-recovery-max-attempts

| | |
|---|---|
| Option | `--cluster-slave-auto-recovery-max-attempts` [398] |
| Aliases | `--cluster-slave-repl-auto-recovery-max-attempts` [398] |
| Config File Options | `cluster-slave-auto-recovery-max-attempts` [398], `cluster-slave-repl-auto-recovery-max-attempts` [398] |
| Description | Default value for --auto-recovery-max-attempts when --topology=cluster-slave |
| Value Type | string |

--cluster-slave-auto-recovery-reset-interval

| | |
|---|---|
| Option | `--cluster-slave-auto-recovery-reset-interval` [398] |

| | |
|---|---|
| Aliases | `--cluster-slave-repl-auto-recovery-reset-interval` [398] |
| Config File Options | `cluster-slave-auto-recovery-reset-interval` [398], `cluster-slave-repl-auto-recovery-reset-interval` [398] |
| Description | Default value for --auto-recovery-reset-interval when --topology=cluster-slave |
| Value Type | string |

--config-file

| | |
|---|---|
| Option | `--config-file` [399] |
| Config File Options | `config-file` [399] |
| Description | Display help information for content of the config file |
| Value Type | string |

--consistency-policy

| | |
|---|---|
| Option | `--consistency-policy` [399] |
| Aliases | `--repl-consistency-policy` [399] |
| Config File Options | `consistency-policy` [399], `repl-consistency-policy` [399] |
| Description | Should the replicator stop or warn if a consistency check fails? |
| Value Type | string |

## 9.8.4. D tpm Options

--dataservice-name

| | |
|---|---|
| Option | `--dataservice-name` [399] |
| Config File Options | `dataservice-name` [399] |
| Description | Limit the command to the hosts in this dataservice Multiple data services may be specified by providing a comma separated list |
| Value Type | string |

--dataservice-relay-enabled

| | |
|---|---|
| Option | `--dataservice-relay-enabled` [399] |
| Config File Options | `dataservice-relay-enabled` [399] |
| Description | Make this dataservice a replica of another |
| Value Type | string |

--dataservice-schema

| | |
|---|---|
| Option | `--dataservice-schema` [399] |
| Config File Options | `dataservice-schema` [399] |
| Description | The db schema to hold dataservice details |
| Value Type | string |

--dataservice-thl-port

| | |
|---|---|
| Option | `--dataservice-thl-port` [399] |
| Config File Options | `dataservice-thl-port` [399] |
| Description | Port to use for THL operations |
| Value Type | numeric |
| Default | 2112 |

--dataservice-vip-enabled

| | |
|---|---|
| Option | `--dataservice-vip-enabled` [399] |

| | |
|---|---|
| Config File Options | dataservice-vip-enabled [399] |
| Description | Is VIP management enabled? |
| Value Type | boolean |

--dataservice-vip-ipaddress

| | |
|---|---|
| Option | --dataservice-vip-ipaddress [400] |
| Config File Options | dataservice-vip-ipaddress [400] |
| Description | VIP IP address |
| Value Type | string |

--dataservice-vip-netmask

| | |
|---|---|
| Option | --dataservice-vip-netmask [400] |
| Config File Options | dataservice-vip-netmask [400] |
| Description | VIP netmask |
| Value Type | string |

--datasource-boot-script

| | |
|---|---|
| Option | --datasource-boot-script [400] |
| Aliases | --repl-datasource-boot-script [400] |
| Config File Options | datasource-boot-script [400], repl-datasource-boot-script [400] |
| Description | Database start script |
| Value Type | string |

--datasource-enable-ssl

| | |
|---|---|
| Option | --datasource-enable-ssl [400] |
| Aliases | --repl-datasource-enable-ssl [400] |
| Config File Options | datasource-enable-ssl [400], repl-datasource-enable-ssl [400] |
| Description | Enable SSL connection to DBMS server |
| Value Type | boolean |

--datasource-group-id

| | |
|---|---|
| Option | --datasource-group-id [400] |
| Config File Options | datasource-group-id [400] |
| Description | All nodes within a cluster that share the same datasource-group-id will form a Distributed Datasource Group (DDG). |
| Value Type | numeric |

All nodes within a cluster that share the same datasource-group-id will form a Distributed Datasource Group (DDG).

--datasource-log-directory

| | |
|---|---|
| Option | --datasource-log-directory [400] |
| Aliases | --repl-datasource-log-directory [400] |
| Config File Options | datasource-log-directory [400], repl-datasource-log-directory [400] |
| Description | Primary log directory |
| Value Type | string |

--datasource-log-pattern

| | |
|---|---|
| Option | --datasource-log-pattern [400] |
| Aliases | --repl-datasource-log-pattern [400] |

| | |
|---|---|
| Config File Options | datasource-log-pattern [400], repl-datasource-log-pattern [400] |
| Description | Primary log filename pattern |
| Value Type | string |

--datasource-mysql-conf

| | |
|---|---|
| Option | --datasource-mysql-conf [401] |
| Aliases | --repl-datasource-mysql-conf [401] |
| Config File Options | datasource-mysql-conf [401], repl-datasource-mysql-conf [401] |
| Description | MySQL config file |
| Value Type | string |

--datasource-mysql-data-directory

| | |
|---|---|
| Option | --datasource-mysql-data-directory [401] |
| Aliases | --repl-datasource-mysql-data-directory [401] |
| Config File Options | datasource-mysql-data-directory [401], repl-datasource-mysql-data-directory [401] |
| Description | MySQL data directory |
| Value Type | string |

--datasource-mysql-ibdata-directory

| | |
|---|---|
| Option | --datasource-mysql-ibdata-directory [401] |
| Aliases | --repl-datasource-mysql-ibdata-directory [401] |
| Config File Options | datasource-mysql-ibdata-directory [401], repl-datasource-mysql-ibdata-directory [401] |
| Description | MySQL InnoDB data directory |
| Value Type | string |

--datasource-mysql-iblog-directory

| | |
|---|---|
| Option | --datasource-mysql-iblog-directory [401] |
| Aliases | --repl-datasource-mysql-iblog-directory [401] |
| Config File Options | datasource-mysql-iblog-directory [401], repl-datasource-mysql-iblog-directory [401] |
| Description | MySQL InnoDB log directory |
| Value Type | string |

--datasource-mysql-ssl-ca

| | |
|---|---|
| Option | --datasource-mysql-ssl-ca [401] |
| Aliases | --repl-datasource-mysql-ssl-ca [401] |
| Config File Options | datasource-mysql-ssl-ca [401], repl-datasource-mysql-ssl-ca [401] |
| Description | MySQL SSL CA file |
| Value Type | string |

--datasource-mysql-ssl-cert

| | |
|---|---|
| Option | --datasource-mysql-ssl-cert [401] |
| Aliases | --repl-datasource-mysql-ssl-cert [401] |
| Config File Options | datasource-mysql-ssl-cert [401], repl-datasource-mysql-ssl-cert [401] |
| Description | MySQL SSL certificate file |
| Value Type | string |

--datasource-mysql-ssl-key

| | |
|---|---|
| Option | `--datasource-mysql-ssl-key` [401] |
| Aliases | `--repl-datasource-mysql-ssl-key` [401] |
| Config File Options | `datasource-mysql-ssl-key` [401], `repl-datasource-mysql-ssl-key` [401] |
| Description | MySQL SSL key file |
| Value Type | string |

--datasource-oracle-service

| | |
|---|---|
| Option | `--datasource-oracle-service` [402] |
| Aliases | `--repl-datasource-oracle-service` [402] |
| Config File Options | `datasource-oracle-service` [402], `repl-datasource-oracle-service` [402] |
| Description | Oracle Service Name |
| Value Type | string |

--datasource-systemctl-service

| | |
|---|---|
| Option | `--datasource-systemctl-service` [402] |
| Aliases | `--repl-datasource-systemctl-service` [402] |
| Config File Options | `datasource-systemctl-service` [402], `repl-datasource-systemctl-service` [402] |
| Description | Database systemctl script |
| Value Type | string |

Specifies the command name or full path of the command that should be used to control the database service, including startup, shutdown and restart. This is used by the Tungsten to control the underlying database service. By default, this will be configured to the service according to your environment if it has been found during installation. For example, the services command, or /etc/init.d/mysql.

--datasource-type

| | | |
|---|---|---|
| Option | `--datasource-type` [402] | |
| Aliases | `--repl-datasource-type` [402] | |
| Config File Options | `datasource-type` [402], `repl-datasource-type` [402] | |
| Description | Database type | |
| Value Type | string | |
| Default | mysql | |
| Valid Values | file | File |
| | hdfs | HDFS (Hadoop) |
| | kafka | Kafka |
| | mongodb | MongoDB |
| | mysql | MySQL |
| | oracle | Oracle |
| | postgres | PostgreSQL |
| | vertica | Vertica |

--delete

| | |
|---|---|
| Option | `--delete` [402] |
| Config File Options | `delete` [402] |
| Description | Delete the named data service from the configuration Data Service options |
| Value Type | string |

--deploy-systemd

| | |
|---|---|
| Option | `--deploy-systemd` [402] |

| | |
|---|---|
| Config File Options | deploy-systemd [402] |
| Description | Setting to true will deploy and configure software to be controlled by systemd. |
| Value Type | string |

--direct-datasource-log-directory

| | |
|---|---|
| Option | --direct-datasource-log-directory [403] |
| Aliases | --repl-direct-datasource-log-directory [403] |
| Config File Options | direct-datasource-log-directory [403], repl-direct-datasource-log-directory [403] |
| Description | Primary log directory |
| Value Type | string |

--direct-datasource-log-pattern

| | |
|---|---|
| Option | --direct-datasource-log-pattern [403] |
| Aliases | --repl-direct-datasource-log-pattern [403] |
| Config File Options | direct-datasource-log-pattern [403], repl-direct-datasource-log-pattern [403] |
| Description | Primary log filename pattern |
| Value Type | string |

--direct-datasource-type

| | | |
|---|---|---|
| Option | --direct-datasource-type [403] | |
| Aliases | --repl-direct-datasource-type [403] | |
| Config File Options | direct-datasource-type [403], repl-direct-datasource-type [403] | |
| Description | Database type | |
| Value Type | string | |
| Default | mysql | |
| Valid Values | file | File |
| | hdfs | HDFS (Hadoop) |
| | mongodb | MongoDB |
| | mysql | |
| | mysql | MySQL |
| | oracle | Oracle |
| | vertica | Vertica |

--direct-replication-host

| | |
|---|---|
| Option | --direct-replication-host [403] |
| Aliases | --direct-datasource-host [403], --repl-direct-datasource-host [403] |
| Config File Options | direct-datasource-host [403], direct-replication-host [403], repl-direct-datasource-host [403] |
| Description | Database server hostname |
| Value Type | string |

--direct-replication-password

| | |
|---|---|
| Option | --direct-replication-password [403] |
| Aliases | --direct-datasource-password [403], --repl-direct-datasource-password [403] |
| Config File Options | direct-datasource-password [403], direct-replication-password [403], repl-direct-datasource-password [403] |
| Description | Password for datasource connection |
| Value Type | string |

--direct-replication-port

| | |
|---|---|
| Option | `--direct-replication-port` [403] |
| Aliases | `--direct-datasource-port` [403], `--repl-direct-datasource-port` [403] |
| Config File Options | `direct-datasource-port` [403], `direct-replication-port` [403], `repl-direct-datasource-port` [403] |
| Description | Database server port |
| Value Type | string |

--direct-replication-user

| | |
|---|---|
| Option | `--direct-replication-user` [404] |
| Aliases | `--direct-datasource-user` [404], `--repl-direct-datasource-user` [404] |
| Config File Options | `direct-datasource-user` [404], `direct-replication-user` [404], `repl-direct-datasource-user` [404] |
| Description | Database login for Tungsten |
| Value Type | string |

--directory

| | |
|---|---|
| Option | `--directory` [404] |
| Config File Options | `directory` [404] |
| Description | Set the directory of an existing installation used during fetching an existing configuration |

Set the directory of an existing installation used during fetching an existing configuration

--disable-relay-logs

| | | |
|---|---|---|
| Option | `--disable-relay-logs` [404] | |
| Aliases | `--repl-disable-relay-logs` [404] | |
| Config File Options | `disable-relay-logs` [404], `repl-disable-relay-logs` [404] | |
| Description | Disable the use of relay-logs? | |
| Value Type | boolean | |
| Default | true | |
| Valid Values | false | |
| | true | |

--disable-security-controls

| | | |
|---|---|---|
| Option | `--disable-security-controls` [404] | |
| Config File Options | `disable-security-controls` [404] | |
| Description | Disables all forms of security, including SSL, TLS and authentication | |
| Value Type | string | |
| Valid Values | false | Default in version 7.x |
| | true | Default in version 5.x and 6.x |

--disable-slave-extractor

| | |
|---|---|
| Option | `--disable-slave-extractor` [404] |
| Aliases | `--repl-disable-slave-extractor` [404] |
| Config File Options | `disable-slave-extractor` [404], `repl-disable-slave-extractor` [404] |
| Description | Should replica servers support the primary role? |
| Value Type | string |

--drop-static-columns-in-updates

| | |
|---|---|
| Option | `--drop-static-columns-in-updates` [404] |
| Config File Options | `drop-static-columns-in-updates` [404] |

| | | |
|---|---|---|
| Description | This will modify UPDATE transactions in row-based replication and eliminate any columns that were not modified. | |
| Value Type | boolean | |
| Default | false | |
| Valid Values | false | |
| | true | |

## 9.8.5. E tpm Options

--enable-active-witnesses

| | | |
|---|---|---|
| Option | `--enable-active-witnesses` [405] | |
| Aliases | `--active-witnesses` [405] | |
| Config File Options | `active-witnesses` [405], `enable-active-witnesses` [405] | |
| Description | Enable active witness hosts | |
| Value Type | boolean | |
| Default | false | |
| Valid Values | false | |
| | true | |

--enable-batch-master

| | | |
|---|---|---|
| Option | `--enable-batch-master` [405] | |
| Config File Options | `enable-batch-master` [405] | |
| Description | Enable batch operation for the primary | |
| Value Type | boolean | |
| Default | false | |
| Valid Values | false | |
| | true | |

--enable-batch-service

| | | |
|---|---|---|
| Option | `--enable-batch-service` [405] | |
| Config File Options | `enable-batch-service` [405] | |
| Description | Enables batch mode for a service | |
| Value Type | boolean | |
| Default | false | |
| Valid Values | false | |
| | true | |

This option enables batch mode for a service, which ensures that replication services that are writing to a target database using batch mode in heterogeneous deployments (for example Hadoop, Amazon Redshift or Vertica). Setting this option enables the following settings on each host:

- On a Primary
  - `mysql-use-bytes-for-string` [414] is set to false.
  - `colnames` filter is enabled (in the `binlog-to-q` stage to add column names to the THL information.
  - `pkey` filter is enabled (in the `binlog-to-q` and `q-to-dbms` stage), with the *`addPkeyToInserts`* and *`addColumnsToDeletes`* filter options set to true. This ensures that rows have the right primary key information.
  - `enumtostring` filter is enabled (in the `q-to-thl` stage), to translate `ENUM` values to their string equivalents.
  - `settostring` filter is enabled (in the `q-to-thl` stage), to translate `SET` values to their string equivalents.
- On a Replica

- mysql-use-bytes-for-string [414] is set to true.
- pkey filter is enabled (q-to-dbms stage).

--enable-batch-slave

| | | |
|---|---|---|
| Option | --enable-batch-slave [406] | |
| Config File Options | enable-batch-slave [406] | |
| Description | Enable batch operation for the Replica | |
| Value Type | boolean | |
| Default | false | |
| Valid Values | false | |
| | true | |

--enable-heterogeneous-master

| | | |
|---|---|---|
| Option | --enable-heterogeneous-master [406] | |
| Config File Options | enable-heterogeneous-master [406] | |
| Description | Enable heterogeneous operation for the primary | |
| Value Type | boolean | |
| Default | false | |
| Valid Values | false | |
| | true | |

--enable-heterogeneous-service

| | | |
|---|---|---|
| Option | --enable-heterogeneous-service [406] | |
| Config File Options | enable-heterogeneous-service [406] | |
| Description | Enable heterogeneous operation | |
| Value Type | boolean | |
| Default | false | |
| Valid Values | false | |
| | true | |

- On a Primary
  - --mysql-use-bytes-for-string [414] is set to false.
  - colnames filter is enabled (in the binlog-to-q stage to add column names to the THL information.
  - pkey filter is enabled (in the binlog-to-q and q-to-dbms stage), with the *addPkeyToInserts* and *addColumnsToDeletes* filter options set to false.
  - enumtostring filter is enabled (in the q-to-thl stage), to translate ENUM values to their string equivalents.
  - settostring filter is enabled (in the q-to-thl stage), to translate SET values to their string equivalents.
- On a Replica
  - --mysql-use-bytes-for-string [414] is set to true.
  - pkey filter is enabled (q-to-dbms stage).

--enable-heterogeneous-slave

| | |
|---|---|
| Option | --enable-heterogeneous-slave [406] |
| Config File Options | enable-heterogeneous-slave [406] |
| Description | Enable heterogeneous operation for the replica |
| Value Type | boolean |

| | | |
|---|---|---|
| Default | false | |
| Valid Values | false | |
| | true | |

--enable-jgroups-ssl

| | |
|---|---|
| Option | `--enable-jgroups-ssl` [407] |
| Aliases | `--jgroups-ssl` [407] |
| Config File Options | `enable-jgroups-ssl` [407], `jgroups-ssl` [407] |
| Description | Enable SSL encryption of JGroups communication on this host |
| Value Type | boolean |

--enable-rmi-authentication

| | |
|---|---|
| Option | `--enable-rmi-authentication` [407] |
| Aliases | `--rmi-authentication` [407] |
| Config File Options | `enable-rmi-authentication` [407], `rmi-authentication` [407] |
| Description | Enable RMI authentication for the services running on this host |
| Value Type | boolean |

--enable-rmi-ssl

| | |
|---|---|
| Option | `--enable-rmi-ssl` [407] |
| Aliases | `--rmi-ssl` [407] |
| Config File Options | `enable-rmi-ssl` [407], `rmi-ssl` [407] |
| Description | Enable SSL encryption of RMI communication on this host |
| Value Type | boolean |

--enable-slave-thl-listener

| | |
|---|---|
| Option | `--enable-slave-thl-listener` [407] |
| Aliases | `--repl-enable-slave-thl-listener` [407] |
| Config File Options | `enable-slave-thl-listener` [407], `repl-enable-slave-thl-listener` [407] |
| Description | Should this service allow THL connections? |
| Value Type | boolean |

--enable-sudo-access

| | | |
|---|---|---|
| Option | `--enable-sudo-access` [407] | |
| Aliases | `--root-command-prefix` [407] | |
| Config File Options | `enable-sudo-access` [407], `root-command-prefix` [407] | |
| Description | Run root commands using sudo | |
| Value Type | boolean | |
| Valid Values | false | |
| | true | |

The default behavior of this property is different in certain installs.

For a cluster node install that INCLUDES the manager process, AND where the install OS user is NOT root, the default will be `true`

For a Replicator only or Connector only install, the default will be `false`

When set to true, the property has the following effect:

- During staging tpm installs, if the tungsten user is different from the ssh user on remote hosts

- All startup scripts when using systemctl: replicator, connector, manager will call systemctl prefixed with sudo: for example: `sudo -n systemctl start treplicator`
- tprovision script (tps.pl) requires sudo access for mysql and xtrabackup calls
- replicator backup script for xtrabackup and other backup utilities
- check_tungsten.sh utility to call xinetd
- tmonitor starts exporter service with sudo
- manager to restart mysql service when found stopped
- tpm diagnostic operation (tpm diag)

--enable-thl-ssl

| | |
|---|---|
| Option | `--enable-thl-ssl` [408] |
| Aliases | `--repl-enable-thl-ssl` [408], `--thl-ssl` [408] |
| Config File Options | `enable-thl-ssl` [408], `repl-enable-thl-ssl` [408], `thl-ssl` [408] |
| Description | Enable SSL encryption of THL communication for this service |
| Value Type | boolean |

--executable-prefix

| | |
|---|---|
| Option | `--executable-prefix` [408] |
| Config File Options | `executable-prefix` [408] |
| Description | Adds a prefix to command aliases |
| Value Type | string |

When enabled, the supplied prefix is added to each command alias that is generated for a given installation. This enables multiple installations to co-exist and and be accessible through a unique alias. For example, if the executable prefix is configured as `east`, then an alias for the installation to trepctl will be created as east_trepctl.

Alias information for executable prefix data is stored within the `$CONTINUENT_ROOT/share/aliases.sh` file for each installation.

## 9.8.6. F tpm Options

--file-protection-level

| | |
|---|---|
| Option | `--file-protection-level` [408] |
| Config File Options | `file-protection-level` [408] |
| Description | Protection level for Continuent files |
| Value Type | string |

--file-protection-umask

| | |
|---|---|
| Option | `--file-protection-umask` [408] |
| Config File Options | `file-protection-umask` [408] |
| Description | Protection umask for Continuent files |
| Value Type | string |

## 9.8.7. H tpm Options

--host-name

| | |
|---|---|
| Option | `--host-name` [408] |
| Config File Options | `host-name` [408] |
| Description | DNS hostname |
| Value Type | string |

--hosts

| | |
|---|---|
| Option | `--hosts` [409] |
| Config File Options | `hosts` [409] |
| Description | Limit the command to the hosts listed You must use the hostname as it appears in the configuration. |
| Value Type | string |

--hub

| | |
|---|---|
| Option | `--hub` [409] |
| Aliases | `--dataservice-hub-host` [409] |
| Config File Options | `dataservice-hub-host` [409], `hub` [409] |
| Description | What is the hub host for this all-masters dataservice? |
| Value Type | string |

--hub-service

| | |
|---|---|
| Option | `--hub-service` [409] |
| Aliases | `--dataservice-hub-service` [409] |
| Config File Options | `dataservice-hub-service` [409], `hub-service` [409] |
| Description | The data service to use for the hub of a star topology |
| Value Type | string |

## 9.8.8. I tpm Options

--install

| | |
|---|---|
| Option | `--install` [409] |
| Config File Options | `install` [409] |
| Description | Install service start scripts |
| Value Type | string |

--install-directory

| | |
|---|---|
| Option | `--install-directory` [409] |
| Aliases | `--home-directory` [409] |
| Config File Options | `home-directory` [409], `install-directory` [409] |
| Description | Installation directory |
| Value Type | string |

Path to the directory where the active deployment will be installed. The configured directory will contain the software, THL and relay log information unless configured otherwise.

## 9.8.9. J tpm Options

--java-enable-concurrent-gc

| | |
|---|---|
| Option | `--java-enable-concurrent-gc` [409] |
| Aliases | `--repl-java-enable-concurrent-gc` [409] |
| Config File Options | `java-enable-concurrent-gc` [409], `repl-java-enable-concurrent-gc` [409] |
| Description | Replicator Java uses concurrent garbage collection |
| Value Type | boolean |

--java-external-lib-dir

| | |
|---|---|
| Option | `--java-external-lib-dir` [409] |
| Aliases | `--repl-java-external-lib-dir` [409] |
| Config File Options | `java-external-lib-dir` [409], `repl-java-external-lib-dir` [409] |
| Description | Directory for 3rd party Jar files required by replicator |
| Value Type | string |

--java-file-encoding

| | |
|---|---|
| Option | `--java-file-encoding` [410] |
| Aliases | `--repl-java-file-encoding` [410] |
| Config File Options | `java-file-encoding` [410], `repl-java-file-encoding` [410] |
| Description | Java platform charset (esp. for heterogeneous replication) |
| Value Type | string |

--java-jgroups-key

| | |
|---|---|
| Option | `--java-jgroups-key` [410] |
| Config File Options | `java-jgroups-key` [410] |
| Description | The alias to use for the JGroups TLS key in the keystore. |
| Value Type | string |

--java-jgroups-keystore-path

| | |
|---|---|
| Option | `--java-jgroups-keystore-path` [410] |
| Config File Options | `java-jgroups-keystore-path` [410] |
| Description | Local path to the JGroups Java Keystore file. |
| Value Type | filename |

--java-jmxremote-access-path

| | |
|---|---|
| Option | `--java-jmxremote-access-path` [410] |
| Config File Options | `java-jmxremote-access-path` [410] |
| Description | Local path to the Java JMX Remote Access file. |
| Value Type | filename |

--java-keystore-password

| | |
|---|---|
| Option | `--java-keystore-password` [410] |
| Config File Options | `java-keystore-password` [410] |
| Description | Set the password for unlocking the tungsten_keystore.jks file in the security directory. Specific for intra cluster communication. |
| Value Type | string |

--java-keystore-path

| | |
|---|---|
| Option | `--java-keystore-path` [410] |
| Config File Options | `java-keystore-path` [410] |
| Description | Local path to the Java Keystore file. Specific for intra cluster communication. NOTE: When java-keystore-path is passed to tpm, the keystore must contain both tls and mysql certs when appropriate. tpm will NOT add mysql cert nor generate tls cert when this flag is found, so both certs must be manually imported already. |
| Value Type | filename |

--java-mem-size

| | |
|---|---|
| Option | `--java-mem-size` [410] |

| | |
|---|---|
| Aliases | `--repl-java-mem-size` [410] |
| Config File Options | `java-mem-size` [410], `repl-java-mem-size` [410] |
| Description | Replicator Java heap memory size in Mb (min 128) |
| Value Type | numeric |

--java-passwordstore-path

| | |
|---|---|
| Option | `--java-passwordstore-path` [411] |
| Config File Options | `java-passwordstore-path` [411] |
| Description | Local path to the Java Password Store file. |
| Value Type | filename |

--java-tls-alias

| | |
|---|---|
| Option | `--java-tls-alias` [411] |
| Config File Options | `java-tls-alias` [411] |
| Description | The alias to use for the TLS key/certificate in the keystore and truststore. |
| Value Type | string |

--java-tls-key-lifetime

| | |
|---|---|
| Option | `--java-tls-key-lifetime` [411] |
| Config File Options | `java-tls-key-lifetime` [411] |
| Description | Lifetime for the Java TLS key |
| Value Type | numeric |

--java-tls-keystore-path

| | |
|---|---|
| Option | `--java-tls-keystore-path` [411] |
| Config File Options | `java-tls-keystore-path` [411] |
| Description | The keystore holding a certificate to use for all Continuent TLS encryption. |
| Value Type | string |

--java-truststore-password

| | |
|---|---|
| Option | `--java-truststore-password` [411] |
| Config File Options | `java-truststore-password` [411] |
| Description | The password for unlocking the tungsten_truststore.jks file in the security directory |
| Value Type | string |

--java-truststore-path

| | |
|---|---|
| Option | `--java-truststore-path` [411] |
| Config File Options | `java-truststore-path` [411] |
| Description | Local path to the Java Truststore file. |
| Value Type | filename |

--java-user-timezone

| | |
|---|---|
| Option | `--java-user-timezone` [411] |
| Aliases | `--repl-java-user-timezone` [411] |
| Config File Options | `java-user-timezone` [411], `repl-java-user-timezone` [411] |
| Description | Java VM Timezone (esp. for cross-site replication) |
| Value Type | numeric |

## 9.8.10. L tpm Options

--log

| | |
|---|---|
| Option | `--log` [412] |
| Config File Options | `log` [412] |
| Description | Write all messages, visible and hidden, to this file. You may specify a filename, 'pid' or 'timestamp'. |
| Value Type | numeric |

--log-slave-updates

| | | |
|---|---|---|
| Option | `--log-slave-updates` [412] | |
| Config File Options | `log-slave-updates` [412] | |
| Description | Should replicas log updates to binlog | |
| Value Type | boolean | |
| Default | false | |
| Valid Values | false | |
| | true | |

## 9.8.11. M tpm Options

--master

| | |
|---|---|
| Option | `--master` [412] |
| Aliases | `--dataservice-master-host` [412], `--masters` [412], `--relay` [412] |
| Config File Options | `dataservice-master-host` [412], `master` [412], `masters` [412], `relay` [412] |
| Description | Hostname of the primary (or relay) host within this service |
| Value Type | string |

The hostname of the primary (extractor) within the current service.

--master-preferred-role

| | | |
|---|---|---|
| Option | `--master-preferred-role` [412] | |
| Aliases | `--repl-master-preferred-role` [412] | |
| Config File Options | `master-preferred-role` [412], `repl-master-preferred-role` [412] | |
| Description | Preferred role for primary THL when connecting as a replica | |
| Value Type | string | |
| Valid Values | master | Primary role |
| | slave | Replica role |

--master-services

| | |
|---|---|
| Option | `--master-services` [412] |
| Aliases | `--dataservice-master-services` [412] |
| Config File Options | `dataservice-master-services` [412], `master-services` [412] |
| Description | Data service names that should be used on each Primary |
| Value Type | string |

--master-thl-host

| | |
|---|---|
| Option | `--master-thl-host` [412] |
| Config File Options | `master-thl-host` [412] |
| Description | Primary THL Hostname |

| | |
|---|---|
| Value Type | string |

--master-thl-port

| | |
|---|---|
| Option | `--master-thl-port` [413] |
| Config File Options | `master-thl-port` [413] |
| Description | Primary THL Port |
| Value Type | string |

--members

| | |
|---|---|
| Option | `--members` [413] |
| Aliases | `--dataservice-hosts` [413] |
| Config File Options | `dataservice-hosts` [413], `members` [413] |
| Description | Hostnames for the dataservice members |
| Value Type | string |

--metadata-directory

| | |
|---|---|
| Option | `--metadata-directory` [413] |
| Aliases | `--repl-metadata-directory` [413] |
| Config File Options | `metadata-directory` [413], `repl-metadata-directory` [413] |
| Description | Replicator metadata directory |
| Value Type | string |

--mysql-allow-intensive-checks

| | | |
|---|---|---|
| Option | `--mysql-allow-intensive-checks` [413] | |
| Config File Options | `mysql-allow-intensive-checks` [413] | |
| Description | For MySQL installation, enables detailed checks on the supported data types within the MySQL database to confirm compatibility. | |
| Value Type | boolean | |
| Default | false | |
| Valid Values | false | |
| | true | |

For MySQL installation, enables detailed checks on the supported data types within the MySQL database to confirm compatibility. This includes checking each table definition individually for any unsupported data types.

--mysql-driver

| | |
|---|---|
| Option | `--mysql-driver` [413] |
| Config File Options | `mysql-driver` [413] |
| Description | MySQL Driver Vendor |
| Value Type | string |
| Default | drizzle |

--mysql-enable-ansiquotes

| | |
|---|---|
| Option | `--mysql-enable-ansiquotes` [413] |
| Aliases | `--repl-mysql-enable-ansiquotes` [413] |
| Config File Options | `mysql-enable-ansiquotes` [413], `repl-mysql-enable-ansiquotes` [413] |
| Description | Enables ANSI_QUOTES mode for incoming events? |
| Value Type | boolean |

--mysql-enable-enumtostring

| | |
|---|---|
| Option | `--mysql-enable-enumtostring` [414] |
| Aliases | `--repl-mysql-enable-enumtostring` [414] |
| Config File Options | `mysql-enable-enumtostring` [414], `repl-mysql-enable-enumtostring` [414] |
| Description | Enable a filter to convert ENUM values to strings |
| Value Type | boolean |

--mysql-enable-noonlykeywords

| | |
|---|---|
| Option | `--mysql-enable-noonlykeywords` [414] |
| Aliases | `--repl-mysql-enable-noonlykeywords` [414] |
| Config File Options | `mysql-enable-noonlykeywords` [414], `repl-mysql-enable-noonlykeywords` [414] |
| Description | Enables a filter to translate `DELETE FROM ONLY` to `DELETE FROM` and `UPDATE ONLY` to `UPDATE`. |
| Value Type | boolean |

--mysql-enable-settostring

| | |
|---|---|
| Option | `--mysql-enable-settostring` [414] |
| Aliases | `--repl-mysql-enable-settostring` [414] |
| Config File Options | `mysql-enable-settostring` [414], `repl-mysql-enable-settostring` [414] |
| Description | Enable a filter to convert SET types to strings |
| Value Type | boolean |

--mysql-ro-slave

| | |
|---|---|
| Option | `--mysql-ro-slave` [414] |
| Aliases | `--repl-mysql-ro-slave` [414] |
| Config File Options | `mysql-ro-slave` [414], `repl-mysql-ro-slave` [414] |
| Description | Replicas are read-only? |
| Value Type | boolean |

--mysql-server-id

| | |
|---|---|
| Option | `--mysql-server-id` [414] |
| Aliases | `--repl-mysql-server-id` [414] |
| Config File Options | `mysql-server-id` [414], `repl-mysql-server-id` [414] |
| Description | Explicitly set the MySQL server ID |
| Value Type | numeric |

Setting this option explicitly sets the server-id information normally located in the MySQL configuration (`my.cnf`). This is useful in situations where there may be multiple MySQL installations and the server ID needs to be identified to prevent collisions when reading from the same Primary.

--mysql-use-bytes-for-string

| | |
|---|---|
| Option | `--mysql-use-bytes-for-string` [414] |
| Aliases | `--repl-mysql-use-bytes-for-string` [414] |
| Config File Options | `mysql-use-bytes-for-string` [414], `repl-mysql-use-bytes-for-string` [414] |
| Description | Transfer strings as their byte representation? |
| Value Type | boolean |

--mysql-xtrabackup-dir

| | |
|---|---|
| Option | `--mysql-xtrabackup-dir` [414] |

| | |
|---|---|
| Aliases | `--repl-mysql-xtrabackup-dir` [414] |
| Config File Options | `mysql-xtrabackup-dir` [414], `repl-mysql-xtrabackup-dir` [414] |
| Description | Directory to use for storing xtrabackup full & incremental backups |
| Value Type | string |

## 9.8.12. N tpm Options

--native-slave-takeover

| | |
|---|---|
| Option | `--native-slave-takeover` [415] |
| Aliases | `--repl-native-slave-takeover` [415] |
| Config File Options | `native-slave-takeover` [415], `repl-native-slave-takeover` [415] |
| Description | Takeover native replication |
| Value Type | boolean |

--no-connectors

| | |
|---|---|
| Option | `--no-connectors` [415] |
| Config File Options | `no-connectors` [415] |
| Description | When issued during an update, connectors will not be restarted. Restart of the connectors will then need to be performed mannually for updates to take affect. |
| Value Type | string |

--no-deployment

| | |
|---|---|
| Option | `--no-deployment` [415] |
| Config File Options | `no-deployment` [415] |
| Description | Skip deployment steps that create the install directory |
| Value Type | string |

--no-validation

| | |
|---|---|
| Option | `--no-validation` [415] |
| Config File Options | `no-validation` [415] |
| Description | Skip validation checks that run on each host |
| Value Type | string |

## 9.8.13. O tpm Options

--optimize-row-events

| | | |
|---|---|---|
| Option | `--optimize-row-events` [415] | |
| Config File Options | `optimize-row-events` [415] | |
| Description | Enables or disables optimized row updates. Enabled by default. | |
| Value Type | boolean | |
| Default | true | |
| Valid Values | false | Disable |

Bundles multiple row-based events into a single `INSERT` or `DELETE` statement. This increases the throughput of large batches of row-based events.

--optimize-row-events-limit-delete-rows

| | |
|---|---|
| Option | `--optimize-row-events-limit-delete-rows` [415] |
| Config File Options | `optimize-row-events-limit-delete-rows` [415] |

| | |
|---|---|
| Description | Limits the number of deletes grouped per event when optimize-row-events is enabled. Note that deletes can only be optimized for tables with single column PK's |
| Value Type | boolean |
| Default | 100 |

--optimize-row-events-limit-insert-rows

| | |
|---|---|
| Option | `--optimize-row-events-limit-insert-rows` [416] |
| Config File Options | `optimize-row-events-limit-insert-rows` [416] |
| Description | Limits the number of inserts grouped per event when optimize-row-events is enabled. |
| Value Type | boolean |
| Default | 50 |

## 9.8.14. P tpm Options

--postgresql-dbname

| | |
|---|---|
| Option | `--postgresql-dbname` [416] |
| Aliases | `--repl-postgresql-dbname` [416] |
| Config File Options | `postgresql-dbname` [416], `repl-postgresql-dbname` [416] |
| Description | Name of the database to replicate |
| Value Type | string |

--prefer-ip-stack

| | | |
|---|---|---|
| Option | `--prefer-ip-stack` [416] | |
| Config File Options | `prefer-ip-stack` [416] | |
| Description | Switch between IPv4 and IPv6 support. | |
| Value Type | string | |
| Default | "4" | |
| Valid Values | "4" | |
| | "6" | |

--preferred-path

| | |
|---|---|
| Option | `--preferred-path` [416] |
| Config File Options | `preferred-path` [416] |
| Description | Additional command path |
| Value Type | filename |

Specifies one or more additional directories that will be added before the current `PATH` environment variable when external commands are run from within the backup environment. This affects all external tools used by Tungsten Cluster, including MySQL, Ruby, Java, and back-up/restore tools such as Percona Xtrabackup.

One or more paths can be specified by separating each directory with a colon. For example:

```
shell> tpm ... --preferred-path=/usr/local/bin:/opt/bin:/opt/percona/bin
```

The `--preferred-path` [416] information propagated to all remote servers within the tpm configuration. However, if the staging server is one of the servers to which you are deploying, the `PATH` must be manually updated.

--prefetch-enabled

| | |
|---|---|
| Option | `--prefetch-enabled` [416] |
| Config File Options | `prefetch-enabled` [416] |
| Description | Should the replicator service be setup as a prefetch applier |

| | |
|---|---|
| Value Type | boolean |

--prefetch-max-time-ahead

| | |
|---|---|
| Option | `--prefetch-max-time-ahead` [417] |
| Config File Options | `prefetch-max-time-ahead` [417] |
| Description | Maximum number of seconds that the prefetch applier can get in front of the standard applier |
| Value Type | numeric |

--prefetch-min-time-ahead

| | |
|---|---|
| Option | `--prefetch-min-time-ahead` [417] |
| Config File Options | `prefetch-min-time-ahead` [417] |
| Description | Minimum number of seconds that the prefetch applier must be in front of the standard applier |
| Value Type | numeric |

--prefetch-schema

| | |
|---|---|
| Option | `--prefetch-schema` [417] |
| Config File Options | `prefetch-schema` [417] |
| Description | Schema to watch for timing prefetch progress |
| Value Type | string |
| Default | tungsten_ |

--prefetch-sleep-time

| | |
|---|---|
| Option | `--prefetch-sleep-time` [417] |
| Config File Options | `prefetch-sleep-time` [417] |
| Description | How long to wait when the prefetch applier gets too far ahead |
| Value Type | string |

--privileged-master

| | |
|---|---|
| Option | `--privileged-master` [417] |
| Config File Options | `privileged-master` [417] |
| Description | Does the login for the Primary database service have superuser privileges |
| Value Type | boolean |

--privileged-slave

| | |
|---|---|
| Option | `--privileged-slave` [417] |
| Config File Options | `privileged-slave` [417] |
| Description | Does the login for the Replica database service have superuser privileges |
| Value Type | booleam |

--profile-script

| | |
|---|---|
| Option | `--profile-script` [417] |
| Config File Options | `profile-script` [417] |
| Description | Append commands to include env.sh in this profile script |
| Value Type | string |

--protect-configuration-files

| | |
|---|---|
| Option | `--protect-configuration-files` [417] |

| | | |
|---|---|---|
| Config File Options | protect-configuration-files [417] | |
| Description | When enabled, configuration files are protected to be only readable and updatable by the configured user | |
| Value Type | string | |
| Valid Values | false | Make configuration files readable by any user |
| | true | |

When enabled (default), the configuration that contain user, password and other information are configured so that they are only readable by the configured user. For example:

```
shell> ls -al /opt/continuent/tungsten/tungsten-replicator/conf/
total 148
drwxr-xr-x 2 tungsten mysql 4096 May 14 14:32 ./
drwxr-xr-x 11 tungsten mysql 4096 May 14 14:32 ../
-rw-r--r-- 1 tungsten mysql 33 May 14 14:32 dynamic-alpha.role
-rw-r--r-- 1 tungsten mysql 5059 May 14 14:32 log4j.properties
-rw-r--r-- 1 tungsten mysql 3488 May 14 14:32 log4j-thl.properties
-rw-r--r-- 1 tungsten mysql 972 May 14 14:32 mysql-java-charsets.properties
-rw-r--r-- 1 tungsten mysql 420 May 14 14:32 replicator.service.properties
-rw-r----- 1 tungsten mysql 1590 May 14 14:35 services.properties
-rw-r----- 1 tungsten mysql 1590 May 14 14:35 .services.properties.orig
-rw-r--r-- 1 tungsten mysql 896 May 14 14:32 shard.list
-rw-r----- 1 tungsten mysql 43842 May 14 14:35 static-alpha.properties
-rw-r----- 1 tungsten mysql 43842 May 14 14:35 .static-alpha.properties.orig
-rw-r----- 1 tungsten mysql 5667 May 14 14:35 wrapper.conf
-rw-r----- 1 tungsten mysql 5667 May 14 14:35 .wrapper.conf.orig
```

When disabled, the files are readable by all users:

```
shell> ll /opt/continuent/tungsten/tungsten-replicator/conf/
total 148
drwxr-xr-x 2 tungsten mysql 4096 May 14 14:32 ./
drwxr-xr-x 11 tungsten mysql 4096 May 14 14:32 ../
-rw-r--r-- 1 tungsten mysql 33 May 14 14:32 dynamic-alpha.role
-rw-r--r-- 1 tungsten mysql 5059 May 14 14:32 log4j.properties
-rw-r--r-- 1 tungsten mysql 3488 May 14 14:32 log4j-thl.properties
-rw-r--r-- 1 tungsten mysql 972 May 14 14:32 mysql-java-charsets.properties
-rw-r--r-- 1 tungsten mysql 420 May 14 14:32 replicator.service.properties
-rw-r--r-- 1 tungsten mysql 1590 May 14 14:32 services.properties
-rw-r--r-- 1 tungsten mysql 1590 May 14 14:32 .services.properties.orig
-rw-r--r-- 1 tungsten mysql 896 May 14 14:32 shard.list
-rw-r--r-- 1 tungsten mysql 43842 May 14 14:32 static-alpha.properties
-rw-r--r-- 1 tungsten mysql 43842 May 14 14:32 .static-alpha.properties.orig
-rw-r--r-- 1 tungsten mysql 5667 May 14 14:32 wrapper.conf
-rw-r--r-- 1 tungsten mysql 5667 May 14 14:32 .wrapper.conf.orig
```

## 9.8.15. R tpm Options

--redshift-dbname

| | |
|---|---|
| Option | --redshift-dbname [418] |
| Aliases | --repl-redshift-dbname [418] |
| Config File Options | redshift-dbname [418], repl-redshift-dbname [418] |
| Description | Name of the Redshift database to replicate into |
| Value Type | string |

--relay-directory

| | |
|---|---|
| Option | --relay-directory [418] |
| Aliases | --repl-relay-directory [418] |
| Config File Options | relay-directory [418], repl-relay-directory [418] |
| Description | Directory for logs transferred from the Primary |
| Value Type | string |
| Default | {home directory}/relay |

--relay-enabled

| | |
|---|---|
| Option | --relay-enabled [418] |

| | |
|---|---|
| Config File Options | relay-enabled [418] |
| Description | Should the replicator service be setup as a relay. |
| Value Type | boolean |

--relay-source

| | |
|---|---|
| Option | --relay-source [419] |
| Aliases | --dataservice-relay-source [419], --master-dataservice [419] |
| Config File Options | dataservice-relay-source [419], master-dataservice [419], relay-source [419] |
| Description | Dataservice name to use as a relay source |
| Value Type | string |

--repl-allow-bidi-unsafe

| | | |
|---|---|---|
| Option | --repl-allow-bidi-unsafe [419] | |
| Config File Options | repl-allow-bidi-unsafe [419] | |
| Description | Allow unsafe SQL from remote service | |
| Value Type | boolean | |
| Default | false | |
| Valid Values | false | |
| | true | |

--repl-store-thl-compressed

| | | |
|---|---|---|
| Option | --repl-store-thl-compressed [419] | |
| Config File Options | repl-store-thl-compressed [419] | |
| Description | Enable (true) or disable (false) compression of THL on disk. | |
| Value Type | boolean | |
| Default | false | |
| Valid Values | false | Disabled |
| | true | Enabled |

Enable (true) or disable (false) compression of THL on disk.

--repl-store-thl-encrypted

| | | |
|---|---|---|
| Option | --repl-store-thl-encrypted [419] | |
| Config File Options | repl-store-thl-encrypted [419] | |
| Description | Enable (true) or disable (false) encryption of THL on disk. | |
| Value Type | boolean | |
| Default | false | |
| Valid Values | false | Disabled |
| | true | Enabled |

Enable (true) or disable (false) encryption of THL on disk.

--repl-svc-extractor-multi-frag-service-detection

| | |
|---|---|
| Option | --repl-svc-extractor-multi-frag-service-detection [419] |
| Config File Options | repl-svc-extractor-multi-frag-service-detection [419] |
| Description | Force extraction to read ahead to last fragment to detect service name |
| Value Type | boolean |
| Default | false |

| | | |
|---|---|---|
| Valid Values | true | Enabled |

When working with multi-active topologies and unprivileged database access (Such as Aurora) the replicator will need to read ahead to the last fragment to ensure multi-fragmented events are correctly flagged by the BidiRemoteSlaveFilter. Disabled by default (false). Set to true to enable.

--repl-thl-client-serialization

| | | |
|---|---|---|
| Option | `--repl-thl-client-serialization` [420] | |
| Config File Options | `repl-thl-client-serialization` [420] | |
| Description | Enable THL compression on downstream Replicator Appliers. | |
| Value Type | string | |
| Default | LEGACY | |
| Valid Values | DEFLATE | |
| | JAVA | |
| | PROTOBUF | |

--repl-thl-server-serialization

| | | |
|---|---|---|
| Option | `--repl-thl-server-serialization` [420] | |
| Config File Options | `repl-thl-server-serialization` [420] | |
| Description | Comma Separated list of THL compression protocols to enable on the thl-server (Extractor). | |
| Value Type | string | |
| Default | LEGACY,JAVA,PROTOBUF,DEFALTE | |
| Valid Values | DEFLATE | |
| | JAVA | |
| | PROTOBUF | |

--replace-tls-certificate

| | |
|---|---|
| Option | `--replace-tls-certificate` [420] |
| Config File Options | `replace-tls-certificate` [420] |
| Description | Replace the TLS certificate |

Replace the TLS certificate

--replication-host

| | |
|---|---|
| Option | `--replication-host` [420] |
| Aliases | `--datasource-host` [420], `--repl-datasource-host` [420] |
| Config File Options | `datasource-host` [420], `repl-datasource-host` [420], `replication-host` [420] |
| Description | Hostname of the datasource |
| Value Type | string |

Hostname of the datasource where the database is located. If the specified hostname matches the current host or member name, the database is assumed to be local. If the hostnames do not match, extraction is assumed to be via remote access. For MySQL hosts, this configures a remote replication Replica (relay) connection.

--replication-password

| | |
|---|---|
| Option | `--replication-password` [420] |
| Aliases | `--datasource-password` [420], `--repl-datasource-password` [420] |
| Config File Options | `datasource-password` [420], `repl-datasource-password` [420], `replication-password` [420] |
| Description | Database password |
| Value Type | string |

The password to be used when connecting to the database using the corresponding `--replication-user` [421].

--replication-port

| | | |
|---|---|---|
| Option | `--replication-port` [421] | |
| Aliases | `--datasource-port` [421], `--repl-datasource-port` [421] | |
| Config File Options | `datasource-port` [421], `repl-datasource-port` [421], `replication-port` [421] | |
| Description | Database network port | |
| Value Type | string | |
| Valid Values | 1521 | Oracle Default |
| | 27017 | Kafka Default |
| | 27017 | MongoDB Default |
| | 3306 | MySQL Default |
| | 5432 | PostgreSQL Default |
| | 5433 | Vertica Default |
| | 5439 | Redshift Default |
| | 8020 | HDFS Default |

The network port used to connect to the database server. The default port used depends on the database being configured.

--replication-user

| | |
|---|---|
| Option | `--replication-user` [421] |
| Aliases | `--datasource-user` [421], `--repl-datasource-user` [421] |
| Config File Options | `datasource-user` [421], `repl-datasource-user` [421], `replication-user` [421] |
| Description | User for database connection |
| Value Type | string |

For databases that required authentication, the username to use when connecting to the database using the corresponding connection method (native, JDBC, etc.).

--replicator-rest-api

| | | |
|---|---|---|
| Option | `--replicator-rest-api` [421] | |
| Config File Options | `replicator-rest-api` [421] | |
| Description | Enable (default) or Disable APIv2 | |
| Default | true | |
| Valid Values | false | |
| | true | |

--replicator-rest-api-address

| | |
|---|---|
| Option | `--replicator-rest-api-address` [421] |
| Config File Options | `replicator-rest-api-address` [421] |
| Description | Address for the API to bind too. |
| Default | 127.0.0.1 |

--replicator-rest-api-authentication

| | |
|---|---|
| Option | `--replicator-rest-api-authentication` [421] |
| Config File Options | `replicator-rest-api-authentication` [421] |
| Description | Enforce authentication for the API. |
| Default | true |

--replicator-rest-api-ssl

| | |
|---|---|
| Option | `--replicator-rest-api-ssl` [421] |
| Config File Options | `replicator-rest-api-ssl` [421] |
| Description | Enable SSL for the API. |
| Default | true |

--reset

| | |
|---|---|
| Option | `--reset` [422] |
| Config File Options | `reset` [422] |
| Description | Clear the current configuration before processing any arguments |
| Value Type | string |

For staging configurations, deletes all pre-existing configuration information between updating with the new configuration values.

--rest-api-admin-pass

| | |
|---|---|
| Option | `--rest-api-admin-pass` [422] |
| Aliases | `--rest-api-admin-password` [422] |
| Config File Options | `rest-api-admin-pass` [422], `rest-api-admin-password` [422] |
| Description | Specify the initial Admin User Password for API access. rest-api-admin-password alias only available from version 7.1.2 onwards. |
| Value Type | string |

Optional: Must be specified along with rest-api-admin-user if you wish to access the full API features.

--rest-api-admin-user

| | |
|---|---|
| Option | `--rest-api-admin-user` [422] |
| Config File Options | `rest-api-admin-user` [422] |
| Description | Specify the initial Admin Username for API access |
| Value Type | string |

Optional: Must be specified along with rest-api-admin-pass if you wish to access the full API features and use the Dashboard GUI for cluster installations.

--rmi-port

| | |
|---|---|
| Option | `--rmi-port` [422] |
| Aliases | `--repl-rmi-port` [422] |
| Config File Options | `repl-rmi-port` [422], `rmi-port` [422] |
| Description | Replication RMI listen port |
| Value Type | string |
| Default | 10001 |

--rmi-user

| | |
|---|---|
| Option | `--rmi-user` [422] |
| Config File Options | `rmi-user` [422] |
| Description | The username for RMI authentication |
| Value Type | string |

--role

| | |
|---|---|
| Option | `--role` [422] |
| Aliases | `--repl-role` [422] |
| Config File Options | `repl-role` [422], `role` [422] |

| | | |
|---|---|---|
| Description | What is the replication role for this service? | |
| Value Type | string | |
| Valid Values | master | |
| | relay | |
| | slave | |

## 9.8.16. S tpm Options

--security-directory

| | |
|---|---|
| Option | `--security-directory` [423] |
| Config File Options | `security-directory` [423] |
| Description | Storage directory for the Java security/encryption files |
| Value Type | string |

--service-alias

| | |
|---|---|
| Option | `--service-alias` [423] |
| Aliases | `--dataservice-service-alias` [423] |
| Config File Options | `dataservice-service-alias` [423], `service-alias` [423] |
| Description | Replication alias of this dataservice |
| Value Type | string |

--service-name

| | |
|---|---|
| Option | `--service-name` [423] |
| Config File Options | `service-name` [423] |
| Description | Set the service name |

Set the service name

--service-type

| | | |
|---|---|---|
| Option | `--service-type` [423] | |
| Aliases | `--repl-service-type` [423] | |
| Config File Options | `repl-service-type` [423], `service-type` [423] | |
| Description | What is the replication service type? | |
| Value Type | string | |
| Valid Values | local | |
| | remote | |

--skip-statemap

| | |
|---|---|
| Option | `--skip-statemap` [423] |
| Config File Options | `skip-statemap` [423] |
| Description | Do not copy the cluster-home/conf/statemap.properties from the previous install |
| Value Type | boolean |

--slaves

| | |
|---|---|
| Option | `--slaves` [423] |
| Aliases | `--dataservice-slaves` [423], `--members` [413] |
| Config File Options | `dataservice-slaves` [423], `members` [413], `slaves` [423] |
| Description | What are the Replicas for this dataservice? |

| | |
|---|---|
| Value Type | string |

--start

| | |
|---|---|
| Option | `--start` [424] |
| Config File Options | `start` [424] |
| Description | Start the services after configuration |
| Value Type | string |

--start-and-report

| | |
|---|---|
| Option | `--start-and-report` [424] |
| Config File Options | `start-and-report` [424] |
| Description | Start the services and report out the status after configuration |
| Value Type | string |

--svc-allow-any-remote-service

| | | |
|---|---|---|
| Option | `--svc-allow-any-remote-service` [424] | |
| Aliases | `--repl-svc-allow-any-remote-service` [424] | |
| Config File Options | `repl-svc-allow-any-remote-service` [424], `svc-allow-any-remote-service` [424] | |
| Description | Replicate from any service | |
| Value Type | boolean | |
| Default | false | |
| Valid Values | true | |

--svc-applier-block-commit-interval

| | | |
|---|---|---|
| Option | `--svc-applier-block-commit-interval` [424] | |
| Aliases | `--repl-svc-applier-block-commit-interval` [424] | |
| Config File Options | `repl-svc-applier-block-commit-interval` [424], `svc-applier-block-commit-interval` [424] | |
| Description | Minimum interval between commits | |
| Value Type | string | |
| Valid Values | 0 | When batch service is not enabled |
| | #d | Number of days |
| | #h | Number of hours |
| | #m | Number of minutes |
| | #s | Number of seconds |

--svc-applier-block-commit-size

| | |
|---|---|
| Option | `--svc-applier-block-commit-size` [424] |
| Aliases | `--repl-svc-applier-block-commit-size` [424] |
| Config File Options | `repl-svc-applier-block-commit-size` [424], `svc-applier-block-commit-size` [424] |
| Description | Applier block commit size (min 1) |
| Value Type | numeric |

--svc-applier-filters

| | |
|---|---|
| Option | `--svc-applier-filters` [424] |
| Aliases | `--repl-svc-applier-filters` [424] |
| Config File Options | `repl-svc-applier-filters` [424], `svc-applier-filters` [424] |
| Description | Replication service applier filters |

| | |
|---|---|
| Value Type | string |

--svc-applier-last-applied-write-interval

| | |
|---|---|
| Option | `--svc-applier-last-applied-write-interval` [425] |
| Config File Options | `svc-applier-last-applied-write-interval` [425] |
| Description | Interval (in seconds) to store the last known applied seqno and latency |
| Value Type | string |

--svc-extractor-filters

| | |
|---|---|
| Option | `--svc-extractor-filters` [425] |
| Aliases | `--repl-svc-extractor-filters` [425] |
| Config File Options | `repl-svc-extractor-filters` [425], `svc-extractor-filters` [425] |
| Description | Replication service extractor filters |
| Value Type | string |

--svc-fail-on-zero-row-update

| | | |
|---|---|---|
| Option | `--svc-fail-on-zero-row-update` [425] | |
| Aliases | `--repl-svc-fail-on-zero-row-update` [425] | |
| Config File Options | `repl-svc-fail-on-zero-row-update` [425], `svc-fail-on-zero-row-update` [425] | |
| Description | How should the replicator behave when a Row-Based Replication UPDATE or DELETE does not affect any rows. | |
| Value Type | string | |
| Default | stop | |
| Valid Values | ignore | No warnings in the log file, and replication continues |
| | warn | Log a Warning in the log file, but continue anyway |

### Warning

> From release 7.0.1 the default for this property was changed to `stop`, previously, the default was `warn`.
>
> The change in the default value may cause unexpected behavior in Active/Active topologies due to the Asynchronous nature of replication, however care should be taken if changing back to the original default of `warn`.
>
> If you notice many entries in your replicator logs indicating zero row updates, and these warnings are being ignored, you may encounter data drift.

--svc-parallelization-type

| | | |
|---|---|---|
| Option | `--svc-parallelization-type` [425] | |
| Aliases | `--repl-svc-parallelization-type` [425] | |
| Config File Options | `repl-svc-parallelization-type` [425], `svc-parallelization-type` [425] | |
| Description | Method for implementing parallel apply | |
| Value Type | string | |
| Valid Values | disk | |
| | memory | |
| | none | |

--svc-remote-filters

| | |
|---|---|
| Option | `--svc-remote-filters` [425] |
| Aliases | `--repl-svc-remote-filters` [425] |
| Config File Options | `repl-svc-remote-filters` [425], `svc-remote-filters` [425] |
| Description | Replication service remote download filters |

| | |
|---|---|
| Value Type | string |

--svc-reposition-on-source-id-change

| | |
|---|---|
| Option | `--svc-reposition-on-source-id-change` [426] |
| Aliases | `--repl-svc-reposition-on-source-id-change` [426] |
| Config File Options | `repl-svc-reposition-on-source-id-change` [426], `svc-reposition-on-source-id-change` [426] |
| Description | The Primary will come ONLINE from the current position if the stored source_id does not match the value in the static properties |
| Value Type | string |

--svc-shard-default-db

| | | |
|---|---|---|
| Option | `--svc-shard-default-db` [426] | |
| Aliases | `--repl-svc-shard-default-db` [426] | |
| Config File Options | `repl-svc-shard-default-db` [426], `svc-shard-default-db` [426] | |
| Description | Mode for setting the shard ID from the default db | |
| Value Type | string | |
| Valid Values | relaxed | |
| | stringent | |

--svc-systemd-config-replicator

| | |
|---|---|
| Option | `--svc-systemd-config-replicator` [426] |
| Config File Options | `svc-systemd-config-replicator` [426] |
| Description | Used to provide custom systemd configuration that will be used in place of the default generated on. |
| Value Type | string |

This property can be used to supply an alternative, custom, systemd configuration that will be used in place of the default generated one.

The value of the parameter should be specified as the name and location of the custom script to use instead, for example:

```
svc-systemd-config-replicator=/home/tungsten/replicator.service
```

Where replicator.service file could look something like the following:

```
[Unit]
Description=Tungsten Replicator
After=mysqld.service mysql.service mariadb.service

[Service]
User=tungsten
Type=forking
ExecStart=/opt/continuent/tungsten/tungsten-replicator/bin/replicator start sysd
ExecStop=/opt/continuent/tungsten/tungsten-replicator/bin/replicator stop sysd
ExecStartPre=/opt/my-pre-start-script.sh

[Install]
WantedBy=multi-user.target
```

--svc-table-engine

| | |
|---|---|
| Option | `--svc-table-engine` [426] |
| Aliases | `--repl-svc-table-engine` [426] |
| Config File Options | `repl-svc-table-engine` [426], `svc-table-engine` [426] |
| Description | Replication service table engine |
| Value Type | string |
| Default | innodb |

--svc-thl-filters

| | |
|---|---|
| Option | `--svc-thl-filters` [426] |

| | |
|---|---|
| Aliases | `--repl-svc-thl-filters` [426] |
| Config File Options | `repl-svc-thl-filters` [426], `svc-thl-filters` [426] |
| Description | Replication service THL filters |
| Value Type | string |

## 9.8.17. T tpm Options

--target-dataservice

| | |
|---|---|
| Option | `--target-dataservice` [427] |
| Aliases | `--slave-dataservice` [427] |
| Config File Options | `slave-dataservice` [427], `target-dataservice` [427] |
| Description | Dataservice to use to determine the value of host configuration |
| Value Type | string |

--temp-directory

| | |
|---|---|
| Option | `--temp-directory` [427] |
| Config File Options | `temp-directory` [427] |
| Description | Temporary Directory |
| Value Type | string |

--template-file

| | |
|---|---|
| Option | `--template-file` [427] |
| Config File Options | `template-file` [427] |
| Description | Display the keys that may be used in configuration template files |
| Value Type | string |

--template-search-path

| | |
|---|---|
| Option | `--template-search-path` [427] |
| Config File Options | `template-search-path` [427] |
| Description | Adds a new template search path for configuration file generation |
| Value Type | filename |

--thl-directory

| | | |
|---|---|---|
| Option | `--thl-directory` [427] | |
| Aliases | `--repl-thl-directory` [427] | |
| Config File Options | `repl-thl-directory` [427], `thl-directory` [427] | |
| Description | Replicator log directory | |
| Value Type | string | |
| Default | {home directory}/thl | |
| Valid Values | {home directory}/thl | |

The given value should be the base directory, to which tungsten will add the service name. For example, the following entry in the tungsten.ini:

```
[alpha]
...
...
thl-directory=/drv1/thl
...
```

Would result in the THL being placed in /drv1/thl/alpha

### Note

Update of thl directory is only available when tpm is called from the staging installation directory, NOT from the running directory.

--thl-do-checksum

| | |
|---|---|
| Option | `--thl-do-checksum` [428] |
| Aliases | `--repl-thl-do-checksum` [428] |
| Config File Options | `repl-thl-do-checksum` [428], `thl-do-checksum` [428] |
| Description | Execute checksum operations on THL log files |
| Value Type | string |

--thl-interface

| | |
|---|---|
| Option | `--thl-interface` [428] |
| Aliases | `--repl-thl-interface` [428] |
| Config File Options | `repl-thl-interface` [428], `thl-interface` [428] |
| Description | Listen interface to use for THL operations |
| Value Type | string |

--thl-log-connection-timeout

| | |
|---|---|
| Option | `--thl-log-connection-timeout` [428] |
| Aliases | `--repl-thl-log-connection-timeout` [428] |
| Config File Options | `repl-thl-log-connection-timeout` [428], `thl-log-connection-timeout` [428] |
| Description | Number of seconds to wait for a connection to the THL log |
| Value Type | numeric |

--thl-log-file-size

| | |
|---|---|
| Option | `--thl-log-file-size` [428] |
| Aliases | `--repl-thl-log-file-size` [428] |
| Config File Options | `repl-thl-log-file-size` [428], `thl-log-file-size` [428] |
| Description | File size in bytes for THL disk logs |
| Value Type | numeric |

--thl-log-fsync

| | |
|---|---|
| Option | `--thl-log-fsync` [428] |
| Aliases | `--repl-thl-log-fsync` [428] |
| Config File Options | `repl-thl-log-fsync` [428], `thl-log-fsync` [428] |
| Description | Fsync THL records on commit. More reliable operation but adds latency to replication when using low-performance storage |
| Value Type | string |

--thl-log-retention

| | |
|---|---|
| Option | `--thl-log-retention` [428] |
| Aliases | `--repl-thl-log-retention` [428] |
| Config File Options | `repl-thl-log-retention` [428], `thl-log-retention` [428] |
| Description | How long do you want to keep THL files. |
| Value Type | string |
| Default | 7d |

| | | |
|---|---|---|
| Valid Values | #d | Number of days |
| | #h | Number of hours |
| | #m | Number of minutes |
| | #s | Number of seconds |

--thl-port

| | |
|---|---|
| Option | `--thl-port` [429] |
| Aliases | `--repl-thl-port` [429] |
| Config File Options | `repl-thl-port` [429], `thl-port` [429] |
| Description | Port to use for THL Operations |
| Value Type | numeric |
| Default | 2112 |

--thl-protocol

| | |
|---|---|
| Option | `--thl-protocol` [429] |
| Aliases | `--repl-thl-protocol` [429] |
| Config File Options | `repl-thl-protocol` [429], `thl-protocol` [429] |
| Description | Protocol to use for THL communication with this service |
| Value Type | string |

--topology

| | | |
|---|---|---|
| Option | `--topology` [429] | |
| Aliases | `--dataservice-topology` [429] | |
| Config File Options | `dataservice-topology` [429], `topology` [429] | |
| Description | Replication topology for the dataservice. | |
| Value Type | string | |
| Valid Values | all-masters | |
| | cluster-alias | |
| | cluster-slave | |
| | clustered | |
| | direct | |
| | fan-in | |
| | master-slave | |
| | star | |

--track-schema-changes

| | |
|---|---|
| Option | `--track-schema-changes` [429] |
| Config File Options | `track-schema-changes` [429] |
| Description | This will enable filters that track DDL statements and write the resulting change to files on Replica hosts. The feature is intended for use in some batch deployments. |
| Value Type | string |

## 9.8.18. U tpm Options

--user

| | |
|---|---|
| Option | `--user` [429] |
| Config File Options | `user` [429] |

| | |
|---|---|
| Description | System User |
| Value Type | string |

## 9.8.19. V tpm Options

--vertica-dbname

| | |
|---|---|
| Option | `--vertica-dbname` [430] |
| Aliases | `--repl-vertica-dbname` [430] |
| Config File Options | `repl-vertica-dbname` [430], `vertica-dbname` [430] |
| Description | Name of the database to replicate into |
| Value Type | string |

## 9.8.20. W tpm Options

--witnesses

| | |
|---|---|
| Option | `--witnesses` [430] |
| Aliases | `--dataservice-witnesses` [430] |
| Config File Options | `dataservice-witnesses` [430], `witnesses` [430] |
| Description | Witness hosts for the dataservice |
| Value Type | string |

# Chapter 10. Tungsten REST API (APIv2)

Version 7.0.0 of the Tungsten suite of products introduced the new Public RESTful API (Referred to as APIv2)

APIv2 will allow quick and easy access to monitoring, replicator manipulation and change of configuration for both humans and software.

A number of key development decisions where made to allow the release of the first version of the new API. The choice was made to release with a high level of security by default, and a minimal set of features so we can release quickly. Future releases will build on this initial foundation:

- Security First: the API is only enabled on localhost (127.0.0.1) by default. Additionally, SSL is now enabled by default (in previous releases, SSL was disabled by default).
- User/Password Protection: HTTP Basic Auth is required to access most API calls. RBAC will come in a future version.
- Per-layer API: each component has its own API.
- Per-host instances: Passwords are stored securely on each host. Of course, the Tungsten installation tool (tpm) allows deployment of identical user/password pairs across nodes, as does the Tungsten Dashboard GUI.
- The REST API provides read-only information through HTTP GET calls. Continuent has made the deliberate choice of using only POST actions when they affect the state or configuration of the cluster. The reason why we made that choice is a matter of simplification: there is an overlap between the definitions of PUT and POST, and the decision to chose between the two is often a balance with pros and cons, triggering sterile discussions. Rather that spending time debating and choosing, we decided to offer only one, so you will not find any use of PUT in our list of functions.

For the full developer documentation, listing all available calls, please refer to the following links:

- API Developer Docs
- Replicator Developer Docs

> **Warning**
>
> API calls triggering configuration changes are protected by a flag, `i-am-sure=true`, in order to avoid unwanted, potentially dramatic, configuration changes. This applies to:
>
> - configuration/module/servicesmap
> - reset
> - offline
> - online
> - onhold
> - addDataService
> - addDataSource

## 10.1. Getting Started with Tungsten REST API

### 10.1.1. Configuring the API

#### 10.1.1.1. Network Ports

The Replicator process listens for API calls using the following port by default:

| Port | Component |
|---|---|
| 8097 | Replicator |

Should you wish to choose your own ports for the API, this can be handled by specifying the new ports in the tpm config using the following properties:

```
replicator-rest-api-port=8097
```

### 10.1.1.2. User Management

With our focus on security in the first release of the API, without any user created, only ping and createAdminUser calls will be available. tpm makes it easy to create the administration user at install through the following options:

```
rest-api-admin-user=tungsten
rest-api-admin-pass=secret
```

Once an admin user is created, all other APIs call will be available using basic HTTP authentication with that user/password.

Should you wish to create the user AFTER installation, this can be done via CLI calls on each node, as explained below.

Using CURL commands

The createAdminUser call will allow creation and modification of REST API users.

```
shell> curl -k -H 'Content-type: application/json' --request POST 'https://127.0.0.1:8096/api/v2/createAdminUser?i-am-sure=true' \
> --data-raw '{
>   "payloadType": "credentials",
>   "user":"tungsten",
>   "pass":"security"
> }'

{ "payloadType":"StringPayload",
  "payloadVersion":"1",
  "payload":{
      "value":"https://127.0.0.1:8096/api/v2/user/tungsten"
            }
}
```

Using tapi

The tapi tool can make the creation of the admin user simple and straightforward, the following example shows the use of this approach.

```
shell> tapi --create --host hostnamehere --create-user usernamehere --create-password passwordhere -v
```

#### Important

Note that users created/modified through the curl and tapi calls only apply to the host on which the call was done. The same, identical call, will have to be re-run on each host of the cluster.

In order to make that last requirement clear, the URL parameter `?i-am-sure=true` will need to be passed to createAdminUser when issued via the curl command as per the example above.

#### Important

Usernames must NOT contain any of the following reserved characters: `(space) ! # $ & ' ( ) * + , / : ; = ? @ [ ]`

### 10.1.1.3. SSL/Encryption

SSL is enabled by default and will use the same keystore, certificates and aliases as the cluster TLS ones.

You can specify them with `java-tls-keystore-path` and `java-truststore-path` tpm options, or let tpm generate the certificates for you during installation.

For more detailed information on SSL, see Chapter 6, *Deployment: Security*

### 10.1.1.4. Enabling and Disabling the API

As previously mentioned, the API is enabled by default, listening only on localhost.

To disable the API, you will need to specify the following tpm options:

```
replicator-rest-api=false
```

To change the address that the API is listening on, you need to specify the following tpm options:

```
replicator-rest-api-address=0.0.0.0
```

## 10.1.2. How to Access the API

### 10.1.2.1. CURL calls and Examples

curl is the simplest command line tool to access the API.

Note that in the examples below, we use self signed certificates, which is why the `-k` flag is passed to cURL

Ping command

```
shell> curl -k --request GET 'https://127.0.0.1:8096/api/v2/ping'

{ "payloadType":"PingPayload",
  "payloadVersion":"1",
  "payload":{
      "message":"Ping test",
      "date":"Mon Sep 20 11:48:48 CEST 2021",
      "hostName":"gilmbp-8.local",
      "pid":16743,
      "jvmUptime":152513
      }
}
```

In this example, we sent a simple ping. Result comes as an `HTTP OK` response with a payload containing a string message "ping test", the date of execution, the hostname of the executor and its pid, plus the number of milliseconds elapsed since the java executable was started

Create admin user

```
shell> curl -k -H 'Content-type: application/json' --request POST 'https://127.0.0.1:8096/api/v2/createAdminUser?i-am-sure=true' \
> --data-raw '{
>   "payloadType": "credentials",
>   "user":"tungsten",
>   "pass":"security"
> }'

{ "payloadType":"StringPayload",
  "payloadVersion":"1",
  "payload":{
      "value":"https://127.0.0.1:8096/api/v2/user/tungsten"
            }
}
```

This POST request show how input data can be passed to API calls as a json string.

In return, the createAdminUser function will give a link to the API call allowing you to display details for the user we just created. Let's follow this link:

```
shell> curl -k --request GET 'https://127.0.0.1:8096/api/v2/user/tungsten' -utungsten:security

{ "payloadType":"CredentialsPayload",
  "payloadVersion":"1",
  "payload":{
      "user":"tungsten",
      "pass":"**obfuscated**",
      "access":"full"
            }
}
```

This call shows how the user/password tuple is passed.

The equivalent call using base64 encoded Authorization header, obtained with `echo -ne "tungsten:security" | base64` would look something like the following:

```
shell> curl -k --request GET 'https://127.0.0.1:8096/api/v2/user/tungsten' --header 'Authorization: Basic dHVuZ3N0ZW46c2VjdXJpdHk='

  "payload":{
      "user":"tungsten",
      "pass":"**obfuscated**",
      "access":"full"
            }
}
```

## 10.1.2.2. tapi

tapi is a convenient command line tool developed by Continuent that will ease access to Tungsten components APIs without having to remember full REST call URLs.

Full documentation on tapi can be found here: Section 8.18, "The tapi Command"

## 10.1.2.3. External Tools

PostMan is one of the most popular GUI tools for REST API testing and use

## 10.1.3. Data Structures

### 10.1.3.1. Generic Payloads

Tungsten API defines its own payloads for both inputs and output. The generic structure looks like the following:

```
{
   "payloadType": "TypeOfPayload",
   "payloadVersion": "1",
   "payload":
   {
     "key"="value"
   }
}
```

Where `payloadType` announces the type of data that will be contained in `payload` in the given `payloadVersion`

As an example, a very simple payload is found in the `StringPayload` data structure and only consists in a key/value pair:

```
{
   "payloadType": "StringPayload",
   "payloadVersion": "1",
   "payload":
   {
      "string"="stringvalue"
   }
}
```

### 10.1.3.2. INPUT and OUTPUT payloads

When starting up with a fresh installation of tungsten, if no admin user has been provided to tpm, credentials can be sent to the various functions via the following payload

```
{
    "payloadType": "CredentialsPayload",
    "payload":
    {
     "user":"<user>",
     "pass":"<password>"
    }
}
```

The same payload structure, slightly enriched, will be found in response to listing the user via:

```
{
    "payloadType": "CredentialsPayload",
    "payloadVersion": "1",
    "payload": {
        "user": "tungsten",
        "pass": "<obfuscated>",
        "access": "full"
    }
}
```

Various other payloads used and produced by Tungsten REST API entry point will be found in the detailed technical documentation. Links below:

- API Developer Docs
- Replicator Developer Docs

### 10.1.3.3. TAPI Datastructures

## 10.2. Replicator API Specifics

The Replicator API is enabled by default but only listens to `localhost` connections on port 8097.

The Replicator REST API can be disabled with the tpm flag:

```
replicator-rest-api=false
```

If required, the listen port can be changed using the tpm option:

```
replicator-rest-api-port=8097
```

Exposing the API to a different network address for remote access, like an internal network, consists in changing the "listen address" of the API server. For example, granting access to local 192.168.1.* network would translate to the tpm flag:

```
replicator-rest-api-listen-address=192.168.1.0
```

> **Warning**
>
> Note that exposing the API to a public network can introduce a security breach like brute-force or DDoS attacks exposure.

The listen port can be change with the tpm flag

```
replicator-rest-port=8097
```

The replicator API has two sets of API calls, based either on the whole replicator or on a service of the replicator. A few examples follow and the full replicator specific developer docs can be viewed here

## 10.2.1. Replicator Endpoints

The following endpoints are available for the whole replicator :

- `services`
- `status`
- `version`
- `online`
- `offline`
- `purge`
- `reset`

### 10.2.1.1. services

The replicators existing services can be listed with `GET` request as follows

```
GET 'https://127.0.0.1:8097/api/v2/replicator/services'
```

The output will look like

```
{
  "payloadType": "ServicesPayload",
  "payloadVersion": "1",
  "payload": [
    {
      "serviceType": "unknown",
      "appliedLastSeqno": -1,
      "relativeLatency": -1,
      "role": "slave",
      "appliedLatency": -1,
      "masterConnectUri": "thl://centos1:2112/",
      "started": true,
      "state": "OFFLINE:ERROR",
      "serviceName": "east"
    },
    {
      "serviceType": "local",
      "appliedLastSeqno": 28,
      "relativeLatency": 434041.992,
      "role": "master",
      "appliedLatency": 1.138,
      "masterConnectUri": "thls://localhost:/",
      "started": true,
      "state": "ONLINE",
      "serviceName": "west"
    }
  ]
}
```

### 10.2.1.2. status

The status of the replicator services will be retrieved using a `GET` request as follows

```
GET 'https://127.0.0.1:8097/api/v2/replicator/status'
```

This will show the individual status of all the replicator services, as this example shows:


```
{
  "payloadType": "ReplicatorStatusPayload",
  "payloadVersion": "1",
  "payload": {
    "east": {
      "appliedLastEventId": "NONE",
      "appliedLastSeqno": -1,
      "appliedLatency": -1,
      "autoRecoveryEnabled": false,
      "autoRecoveryTotal": 0,
      "channels": -1,
      "clusterName": "east",
      "currentEventId": "NONE",
      "currentTimeMillis": 1646836106399,
      "dataServerHost": "continuent4",
      "extensions": "",
      "host": "continuent4",
      "latestEpochNumber": -1,
      "masterConnectUri": "thl://centos1:2112/",
      "masterListenUri": "thl://continuent4:2112/",
      "maximumStoredSeqNo": -1,
      "minimumStoredSeqNo": -1,
      "offlineRequests": "NONE",
      "pendingError": "Event application failed: seqno=10 fragno=0 message=Table »
          mats.ct1351 not found in database. Unable to generate a valid statement.",
      "pendingErrorCode": "NONE",
      "pendingErrorEventId": "mysql-bin.000015:0000000000368340;-1",
      "pendingErrorSeqno": 10,
      "pendingExceptionMessage": "Table mats.ct1351 not found in database. Unable »
          to generate a valid statement.",
      "pipelineSource": "UNKNOWN",
      "relativeLatency": -1,
      "resourceJdbcDriver": "org.drizzle.jdbc.DrizzleDriver",
      "resourceJdbcUrl": "jdbc:mysql:thin://continuent4:3306/${DBNAME}? »
          jdbcCompliantTruncation=false&zeroDateTimeBehavior=convertToNull& »
          tinyInt1isBit=false&allowMultiQueries=true&yearIsDateType=false",
      "resourcePrecedence": 99,
      "resourceVendor": "mysql",
      "rmiPort": 10000,
      "role": "slave",
      "seqnoType": "java.lang.Long",
      "serviceName": "east",
      "serviceType": "unknown",
      "simpleServiceName": "east",
      "siteName": "default",
      "sourceId": "continuent4",
      "state": "OFFLINE:ERROR",
      "timeInStateSeconds": 434526.394,
      "timezone": "GMT",
      "transitioningTo": "",
      "uptimeSeconds": 434526.967,
      "useSSLConnection": false,
      "version": "Tungsten Replicator 7.0.0"
    },
    "west": {
      "appliedLastEventId": "mysql-bin.000016:0000000000010112;-1",
      "appliedLastSeqno": 28,
      "appliedLatency": 1.138,
      "autoRecoveryEnabled": false,
      "autoRecoveryTotal": 0,
      "channels": 1,
      "clusterName": "west",
      "currentEventId": "mysql-bin.000016:0000000000010112",
      "currentTimeMillis": 1646836106401,
      "dataServerHost": "continuent4",
      "extensions": "",
      "host": "continuent4",
      "latestEpochNumber": 28,
      "masterConnectUri": "thls://localhost:/",
      "masterListenUri": "thls://continuent4:2112/",
      "maximumStoredSeqNo": 28,
      "minimumStoredSeqNo": 0,
      "offlineRequests": "NONE",
      "pendingError": "NONE",
      "pendingErrorCode": "NONE",
      "pendingErrorEventId": "NONE",
      "pendingErrorSeqno": -1,
      "pendingExceptionMessage": "NONE",
      "pipelineSource": "jdbc:mysql:thin://continuent4:3306/tungsten_west? »
          noPrepStmtCache=true&allowMultiQueries=true&stripQueryComments=false& »
```

```
          enabledProtocols=TLSv1.3,TLSv1.2,TLSv1.1&enabledCipherSuites= »
          TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384, »
          TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA,TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA, »
          TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA,TLS_ECDH_RSA_WITH_AES_256_CBC_SHA, »
          TLS_RSA_WITH_AES_256_CBC_SHA,TLS_RSA_WITH_AES_128_CBC_SHA&connectTimeout=15",
      "relativeLatency": 434526.401,
      "resourceJdbcDriver": "org.drizzle.jdbc.DrizzleDriver",
      "resourceJdbcUrl": "jdbc:mysql:thin://continuent4:3306/${DBNAME}? »
          jdbcCompliantTruncation=false&zeroDateTimeBehavior=convertToNull& »
          tinyInt1isBit=false&allowMultiQueries=true&yearIsDateType=false",
      "resourcePrecedence": 99,
      "resourceVendor": "mysql",
      "rmiPort": 10000,
      "role": "master",
      "seqnoType": "java.lang.Long",
      "serviceName": "west",
      "serviceType": "local",
      "simpleServiceName": "west",
      "siteName": "default",
      "sourceId": "continuent4",
      "state": "ONLINE",
      "timeInStateSeconds": 434526.28,
      "timezone": "GMT",
      "transitioningTo": "",
      "uptimeSeconds": 434526.479,
      "useSSLConnection": true,
      "version": "Tungsten Replicator 7.0.0"
    }
  }
}
```

Optional parameters can be added to specify the kind of status. the following parameters are:

- `channel_assignments`
- `services`
- `shards`
- `stages`
- `stores`
- `tasks`
- `watches`

For example, to view the status of tasks, issue:

```
GET 'https://127.0.0.1:8097/api/v2/replicator/status?statusType=tasks'
```

Which will produce output similar to the following example:

```
{
  "payloadType": "ReplicatorStatusByTypePayload",
  "payloadVersion": "1",
  "payload": {
    "type": "tasks",
    "status": {
      "west": [
        {
          "lastCommittedBlockTime": "1.037",
          "currentLastSeqno": "28",
          "extractTime": "1.034",
          "currentLastEventId": "mysql-bin.000016:0000000000010112;-1",
          "eventCount": "1",
          "filterTime": "0.0",
          "appliedLastSeqno": "28",
          "averageBlockSize": "0.500     ",
          "appliedLastEventId": "mysql-bin.000016:0000000000010112;-1",
          "stage": "binlog-to-q",
          "currentBlockSize": "0",
          "appliedLatency": "1.123",
          "cancelled": "false",
          "commits": "2",
          "otherTime": "0.0",
          "timeInCurrentEvent": "434695.351",
          "currentLastFragno": "0",
          "state": "extract",
          "applyTime": "0.0",
          "taskId": "0",
```

```
          "lastCommittedBlockSize": "1"
        },
        {
          "lastCommittedBlockTime": "1.068",
          "currentLastSeqno": "28",
          "extractTime": "1.034",
          "currentLastEventId": "mysql-bin.000016:0000000000010112;-1",
          "eventCount": "1",
          "filterTime": "0.0",
          "appliedLastSeqno": "28",
          "averageBlockSize": "0.500     ",
          "appliedLastEventId": "mysql-bin.000016:0000000000010112;-1",
          "stage": "q-to-thl",
          "currentBlockSize": "0",
          "appliedLatency": "1.123",
          "cancelled": "false",
          "commits": "2",
          "otherTime": "0.001",
          "timeInCurrentEvent": "434695.336",
          "currentLastFragno": "0",
          "state": "extract",
          "applyTime": "0.015",
          "taskId": "0",
          "lastCommittedBlockSize": "1"
        }
      ],
      "east": []
    }
  }
}
```

### 10.2.1.3. version

This `GET` request returns the running version of the replicator

```
GET 'https://127.0.0.1:8097/api/v2/replicator/version'
```

```
{
  "payloadType": "StringPayload",
  "payloadVersion": "1",
  "payload": {
    "value": "Tungsten Replicator 7.0.0"
  }
}
```

For more precise version of the replicator, another `GET` request can be used

```
GET 'https://127.0.0.1:8097/api/v2/tungstenVersion'
```

```
{
  "payloadType": "VersionPayload",
  "payloadVersion": "1",
  "payload": {
    "productName": "Tungsten Replicator",
    "versionMajor": 7,
    "versionMinor": 0,
    "versionRevision": 0,
    "versionBuild": null,
    "tungstenVersion": "Tungsten Replicator 7.0.0"
  }
}
```

### 10.2.1.4. offline/online

These endpoints allows putting all services online / offline.

This is a `POST` request sent, for example

```
POST 'https://127.0.0.1:8097/api/v2/replicator/online'
POST 'https://127.0.0.1:8097/api/v2/replicator/offline'
```

This will return a task via a `TaskPayload` as below

```
{
  "payloadType": "TaskPayload",
  "payloadVersion": "1",
  "payload": {
    "taskId": "ecaed8a7-5b57-4a76-a1a5-a2de14d9cc79",
    "state": "in_progress",
    "operation": "OnlineTask"
  }
```

```
}
```

Both operations accept an input payload (`OnlinePayload` / `OfflinePayload`). An example is shown below :

```
{
  "payloadType": "OnlinePayload",
  "payloadVersion": "1",
  "payload": {
    "options": [
      {
        "type":"toHeartbeat",
        "value":"stop"
      }
    ]
  }
}
```

### 10.2.1.5. purge

This endpoint purges all non-tungsten connections connected to the different services.

This is a `POST` request as follows

```
POST 'https://127.0.0.1:8097/api/v2/replicator/purge'
```

A `PurgePayload` can be provided to specify a timeout. This will return a `TaskPayload` as an online call, for example.

### 10.2.1.6. reset

This endpoint resets all replicator services.

This is a `POST` request as follows

```
POST 'https://127.0.0.1:8097/api/v2/replicator/reset'
```

A `ResetPayload` can be provided to specify what should be reseted, for exmaple:

```
{
  "payloadType": "ResetPayload",
  "payloadVersion": "1",
  "payload": {
    "type": "thl"
  }
}
```

Valid values for the type are : `all, db, thl, relay`

This will return a `TaskPayload` as an online call, for example.

## 10.2.2. Service Endpoints

Services can also be controlled individually through a number of endpoints. Some are the same as the main replicator service endpoints (`service, status, online, offline, purge, reset`), but other are specific to services, the full list is as follows:

- `backupCapabilities`
- `backups`
- `backup`
- `restore`
- `check`
- `heartbeat`
- `perf`
- `wait`
- `clients`
- `setrole`
- `flush`

- `clear`

A few examples of a selection of these endpoints follow

### 10.2.2.1. backupCapabilities

Returns the defined backup capabilities for the service.

This is a `GET` request to

```
GET 'https://127.0.0.1:8097/api/v2/replicator/service/west/backupCapabilities'
```

and returns a `BackupCapabilitiesPayload`, for exmaple:

```
{
  "payloadType": "BackupCapabilitiesPayload",
  "payloadVersion": "1",
  "payload": {
    "storageAgents": [
      "fs"
    ],
    "backupAgents": [
      "mariabackup-full",
      "mariabackup-incremental",
      "mysqldump",
      "xtrabackup-incremental",
      "xtrabackup-full",
      "mariabackup",
      "xtrabackup"
    ],
    "defaultBackupAgent": "mysqldump",
   "defaultStorageAgent": "fs"
  }
}
```

### 10.2.2.2. backups

This is a `GET` request that returns a list of existing backups for the service.

### 10.2.2.3. backup / restore

Backups or restores the service.

This is a `POST` request :

```
POST 'https://127.0.0.1:8097/api/v2/replicator/service/west/backup'
or
POST 'https://127.0.0.1:8097/api/v2/replicator/service/west/restore'
```

and a `BackupPayLoad` can be specified to provide different settings :

```
{
  "payloadType": "BackupPayload",
  "payloadVersion": "1",
  "payload": {
    "agentName": "mysqldump",
    "storageName": "fs"
  }
}
```

If no payload is provided, backup will use the default backup and storage agents, as shown by `backupCapabilities`, while restore will use the last available backup of the service.

### 10.2.2.4. setrole

Changes the role of the replicator service.

This is a `POST` request

```
POST 'https://127.0.0.1:8097/api/v2/replicator/service/west/setrole'
```

A payload is mandatory to specify which should be the new role, for example:

```
{
  "payloadType": "SetRolePayload",
  "payloadVersion": "1",
```

```
  "payload": {
    "role": "primary"
  }
}
```

Valid roles are : `primary, replica, relay, archive, thl_server, thl_client`

For other calls, refer to the Replicator API Developer documentation.

## 10.2.3. Service THL Endpoints

A few endpoints are provided to get information about the service THL or to manipulate it, these are as follows:

- `index`
- `info`
- `encryption`
- `compression`
- `genkey`
- `index / info / compression / encryption`

These `GET` requests provide information about THL

**`index`** : THL files index

```
GET 'https://127.0.0.1:8097/api/v2/replicator/service/west/thl/index'
```

**`info`** : THL metadata

```
GET 'https://127.0.0.1:8097/api/v2/replicator/service/west/thl/info'
```

**`encryption`** : encryption state of THL (enabled or not)

```
GET 'https://127.0.0.1:8097/api/v2/replicator/service/west/thl/encryption'
```

**`compression`** : compression state of THL (enabled or not)

```
GET 'https://127.0.0.1:8097/api/v2/replicator/service/west/thl/compression'
```

An example follows, for more detail please refer to the Replicator API Developer documentation.

### 10.2.3.1. compression / encryption

When used with a `POST` request, this will turn on or off either compression or encryption.

```
POST 'https://127.0.0.1:8097/api/v2/replicator/service/west/thl/encryption'
```

Here is an example to turn encryption on :

```
{
  "payloadType": "BooleanPayload",
  "payloadVersion": "1",
  "payload": {
    "value": true
  }
}
```

The service has to be offline to turn compression / encryption on or off.

### 10.2.3.2. genkey

This is a `POST` request that will ask the replicator to generate a new encryption key for the THL.

The service has to be online to generate a new THL key.

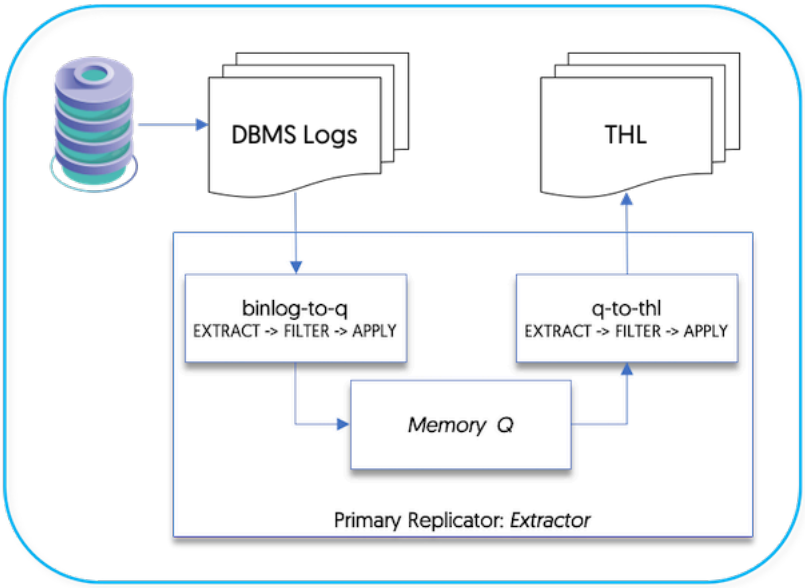
# Chapter 11. Replication Filters

Filtering operates by applying the filter within one, or more, of the stages configured within the replicator. Stages are the individual steps that occur within a pipeline, that take information from a source (such as MySQL binary log) and write that information to an internal queue, the transaction history log, or apply it to a database. Where the filters are applied ultimately affect how the information is stored, used, or represented to the next stage or pipeline in the system.

For example, a filter that removed out all the tables from a specific database would have different effects depending on the stage it was applied. If the filter was applied on the Extractor before writing the information into the THL, then no Applier could ever access the table data, because the information would never be stored into the THL to be transferred to the Targets. However, if the filter was applied on the Applier, then some Appliers could replicate the table and database information, while other Appliers could choose to ignore them. The filtering process also has an impact on other elements of the system. For example, filtering on the Extractor may reduce network overhead, albeit at a reduction in the flexibility of the data transferred.

In a standard replicator configuration with MySQL, the following stages are configured in the Extractor, as shown in Figure 11.1, "Filters: Pipeline Stages on Extractors".

Figure 11.1. Filters: Pipeline Stages on Extractors




Where:

- **`binlog-to-q`** Stage

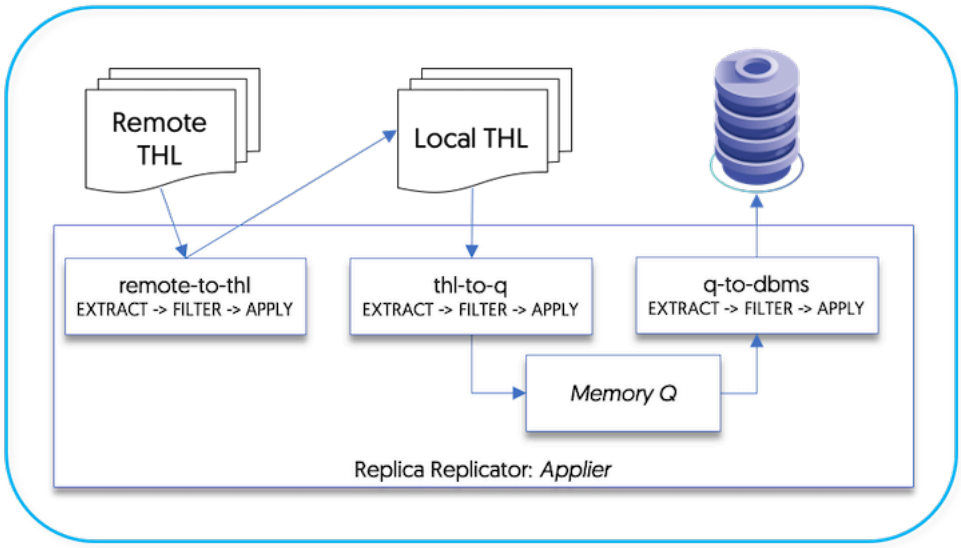
  The `binlog-to-q` stage reads information from the MySQL binary log and stores the information within an in-memory queue.

- **`q-to-thl`** Stage

  The in-memory queue is written out to the THL file on disk.

Within the Applier, the stages configured by default are shown in Figure 11.2, "Filters: Pipeline Stages on Appliers".

Figure 11.2. Filters: Pipeline Stages on Appliers




- `remote-to-thl` Stage

  Remote THL information is read from an trext; datasource and written to a local file on disk.

- `thl-to-q` Stage

  The THL information is read from the file on disk and stored in an in-memory queue.

- `q-to-dbms` Stage

  The data from the in-memory queue is written to the target database.

Filters can be applied during any configured stage, and where the filter is applied, alters the content and availability of the information. The staging and filtering mechanism can also be used to apply multiple filters to the data, altering content when it is read and when it is applied.

Where more than one filter is configured for a pipeline, each filter is executed in the order it appears in the configuration. For example, within the following fragment:

```
...
replicator.stage.binlog-to-q.filters=settostring,enumtostring,pkey,colnames
...
```

`settostring` is executed first, followed by `enumtostring`, `pkey` and finally `colnames`.

For certain filter combinations this order can be significant. Some filters rely on the information provided by earlier filters.

# 11.1. Enabling/Disabling Filters

A number of standard filter configurations are created and defined by default within the static properties file for the Tungsten Replicator configuration.

Filters can be enabled through tpm to update the filter configuration

- `--repl-svc-extractor-filters` [425]

  Apply the filter during the extraction stage, i.e. when the information is extracted from the binary log and written to the internal queue (`binlog-to-q`).

- `--repl-svc-thl-filters` [426]

  Apply the filter between the internal queue and when the transactions are written to the THL on the Extractor. (`q-to-thl`).

- `--repl-svc-remote-filters` [425]

  Apply the filter between reading from the remote THL server and writing to the local THL files on the Applier (`remote-to-thl`).

- `--repl-svc-applier-filters` [424]

  Apply the filter between reading from the internal queue and applying to the destination database (`q-to-dbms`).

Properties and options for an individual filter can be specified by setting the corresponding property value on the tpm command-line.

For example, to ignore a database schema on a Applier, the `replicate` filter can be enabled, and the *`replicator.filter.replicate.ignore`* specifies the name of the schemas to be ignored. To ignore the schema `contacts`:

For staging edeployments:

```
shell> ./tools/tpm update alpha --hosts=host1,host2,host3 \
    --repl-svc-applier-filters=replicate \
    --property=replicator.filter.replicate.ignore=contacts
```

For ini deployments:

```
shell> vi /etc/tungsten/tungsten.ini

[servicename]
...
repl-svc-applier-filters=replicate
property=replicator.filter.replicate.ignore=contacts
...

shell> tpm update
```

A bad filter configuration will not stop the replicator from starting, but the replicator will be placed into the `OFFLINE` [195] state.

To disable a previously enabled filter for staging deployments, empty the filter specification and (optionally) unset the corresponding property or properties. For example:

```
shell> ./tools/tpm update alpha --hosts=host1,host2,host3 \
    --repl-svc-applier-filters= \
    --remove-property=replicator.filter.replicate.ignore
```

To disable a previously enabled filter for ini deployments, remove the values from the `tungsten.ini` file, and issue tpm update

Multiple filters can be applied on any stage, and the filters will be processed and called within the order defined within the configuration. For example, the following configuration:

```
shell> ./tools/tpm update alpha --hosts=host1,host2,host3 \
    --repl-svc-applier-filters=enumtostring,settostring,pkey \
    --remove-property=replicator.filter.replicate.ignore
```

The filters are called in order:

1. `enumtostring`
2. `settostring`
3. `pkey`

The order and sequence can be important if operations are being performed on the data and they are relied on later in the stage. For example, if data is being filtered by a value that exists in a `SET` column within the source data, the `settostring` filter must be defined before the data is filtered, otherwise the actual string value will not be identified.

### Warning

> In some cases, the filter order and sequence can also introduce errors. For example, when using the `pkey` filter and the `optimizeupdates` filters together, `pkey` may remove KEY information from the THL before `optimizeupdates` attempts to optimize the ROW event, causing the filter to raise a failure condition.

The currently active filters can be determined by using the trepctl status -name stages command:

```
shell> trepctl status -name stages
Processing status command (stages)...
...
NAME                 VALUE
----                 -----
applier.class      : com.continuent.tungsten.replicator.applier.MySQLDrizzleApplier
applier.name       : dbms
blockCommitRowCount: 10
committedMinSeqno  : 3600
extractor.class    : com.continuent.tungsten.replicator.thl.THLParallelQueueExtractor
extractor.name     : parallel-q-extractor
```

```
filter.0.class     : com.continuent.tungsten.replicator.filter.MySQLSessionSupportFilter
filter.0.name      : mysqlsessions
filter.1.class     : com.continuent.tungsten.replicator.filter.PrimaryKeyFilter
filter.1.name      : pkey
filter.2.class     : com.continuent.tungsten.replicator.filter.BidiRemoteSlaveFilter
filter.2.name      : bidiSlave
name               : q-to-dbms
processedMinSeqno  : -1
taskCount          : 5
Finished status command (stages)...
```

The above output is from a standard Applier replication installation showing the default filters enabled. The filter order can be determined by the number against each filter definition.

## 11.2. Enabling Additional Filters

The Tungsten Replicator configuration includes a number of filter configurations by default. However, not all filters are given a default configuration, and for some filters, multiple configurations may be needed to achieve more complex filtering requirements. Internally, filter configuration is defined through a property file that defines the filter name and corresponding parameters.

For example, the `rename` configuration is defined as follows:

```
replicator.filter.rename=com.continuent.tungsten.replicator.filter.RenameFilter
replicator.filter.rename.definitionsFile=${replicator.home.dir}/samples/extensions/java/rename.csv
```

The first line creates a new filter configuration using the corresponding Java class. In this case, the filter is named `rename`, as defined by the string *`replicator.filter.rename`*.

Configuration parameters for the filter are defined as values after the filter name. In this example, *`definitionsFile`* is the name of the property examined by the class to set the CSV file where the rename definitions are located.

To create an entirely new filter based on an existing filter class, a new property should created with the new filter definition in the configuration file.

Additional properties from this base should then be used. For example, to create a second rename filter definition called `custom`:

```
replicator.filter.rename.custom=com.continuent.tungsten.replicator.filter.RenameFilter
replicator.filter.rename.custom.definitionsFile=${replicator.home.dir}/samples/extensions/java/renamecustom.csv
```

The filter can be enabled against the desired stage using the filter name `custom`:

```
shell> ./tools/tpm configure \
    --repl-svc-applier-filters=custom
```

## 11.3. Filter Status

To determine which filters are currently being applied within a replicator, use the trepctl status -name stages command. This outputs a list of the current stages and their configuration. For example:

```
shell> trepctl status -name stages
Processing status command (stages)...
NAME                 VALUE
----                 -----
applier.class      : com.continuent.tungsten.replicator.thl.THLStoreApplier
applier.name       : thl-applier
blockCommitRowCount: 1
committedMinSeqno  : 15
extractor.class    : com.continuent.tungsten.replicator.thl.RemoteTHLExtractor
extractor.name     : thl-remote
name               : remote-to-thl
processedMinSeqno  : -1
taskCount          : 1
NAME                 VALUE
----                 -----
applier.class      : com.continuent.tungsten.replicator.thl.THLParallelQueueApplier
applier.name       : parallel-q-applier
blockCommitRowCount: 10
committedMinSeqno  : 15
extractor.class    : com.continuent.tungsten.replicator.thl.THLStoreExtractor
extractor.name     : thl-extractor
name               : thl-to-q
processedMinSeqno  : -1
taskCount          : 1
NAME                 VALUE
----                 -----
applier.class      : com.continuent.tungsten.replicator.applier.MySQLDrizzleApplier
```

```
applier.name       : dbms
blockCommitRowCount: 10
committedMinSeqno  : 15
extractor.class    : com.continuent.tungsten.replicator.thl.THLParallelQueueExtractor
extractor.name     : parallel-q-extractor
filter.0.class     : com.continuent.tungsten.replicator.filter.TimeDelayFilter
filter.0.name      : delay
filter.1.class     : com.continuent.tungsten.replicator.filter.MySQLSessionSupportFilter
filter.1.name      : mysqlsessions
filter.2.class     : com.continuent.tungsten.replicator.filter.PrimaryKeyFilter
filter.2.name      : pkey
name               : q-to-dbms
processedMinSeqno  : -1
taskCount          : 5
Finished status command (stages)...
```

In the output, the filters applied to the applier stage are shown in the last block of output. Filters are listed in the order in which they appear within the configuration.

For information about the filter operation and any modifications or changes made, check the `trepsvc.log` log file.

# 11.4. Filter Reference

The different filter types configured and available within Tungsten Replicator are designed to provide a number of different functionality and operations. Since the information exchanged through the THL system contains a copy of the statement or the row data that is being updated, the filters allow schemas, table and column names, as well as actual data to be converted at the stage in which they are applied.

Filters are identified according to the underlying Java class that defines their operation. For different filters, further configuration and naming is applied according to the templates used when Tungsten Cluster is installed through tpm.

Tungsten Replicator also comes with a number of JavaScript filters that can either be used directly, or that can be modified and adapted to suit individual requirements. These filter scripts are located in `tungsten-replicator/support/filters-javascript`.

For the purposes of classification, the different filters have been categorised according to their main purpose:

- Auditing

  These filters provide methods for tracking database updates alongside the original table data. For example, in a financial database, the actual data has to be updated in the corresponding tables, but the individual changes that lead to that update must also be logged individually.

- Content

  Content filters modify or update the content of the transaction events. These may alter information, for the purposes of interoperability (such as updating enumerated or integer values to their string equivalents), or remove or filter columns, tables, and entire schemas.

- Logging

  Logging filters record information about the transactions into the standard replicator log, either for auditing or debugging purposes.

- Optimization

  The optimization filters are designed to simplify and optimize statements and row updates to improve the speed at which those updates can be applied to the destination dataserver.

- Transformation

  Transformation filters rename or reformat schemas and tables according to a set of rules. For example, multiple schemas can be merged to a single schema, or tables and column names can be updated

- Validation

  Provide validation or consistency checking of either the data or the replication process.

- Miscellaneous

  Other filters that cannot be allocated to one of the existing filter classes.

In the following reference sections:

- Pre-configured filter name is the filter name that can be used against a stage without additional configuration.
- Property prefix is the prefix string for the filter to be used when assigning property values.

- Classname is the Java class name of the filter.
- Parameter is the name of the filter parameter can be set as a property within the configuration.
- Data compatibility indicates whether the filter is compatible with row-based events, statement-based events, or both.

## 11.4.1. `ansiquotes.js` Filter

The `ansiquotes` filter operates by inserting an SQL mode change to `ANSI_QUOTES` into the replication stream before a statement is executed, and returning to an empty SQL mode.

| | | | |
|---|---|---|---|
| Pre-configured filter name | | `ansiquotes` | |
| JavaScript Filter File | | `tungsten-replicator/support/filters-javascript/ansiquotes.js` | |
| Property prefix | | *`replicator.filter.ansiquotes`* | |
| Stage compatibility | | `binlog-to-q` | |
| tpm Option compatibility | | `--svc-extractor-filters` [425] | |
| Data compatibility | | Any event | |
| Parameters | | | |
| Parameter | Type | Default | Description |

This changes a statement such as:

```
INSERT INTO notepad VALUES ('message',0);
```

To:

```
SET sql_mode='ANSI_QUOTES';
INSERT INTO notepad VALUES ('message',0);
SET sql_mode='';
```

This is achieved within the JavaScript by processing the incoming events and adding a new statement before the first `DBMSData` object in each event:

```
query = "SET sql_mode='ANSI_QUOTES'";
newStatement = new com.continuent.tungsten.replicator.dbms.StatementData(
    query,
    null,
    null
    );
data.add(0, newStatement);
```

A corresponding statement is appended to the end of the event:

```
query = "SET sql_mode='''";
newStatement = new com.continuent.tungsten.replicator.dbms.StatementData(
    query,
    null,
    null
    );
data.add(data.size(), newStatement);
```

## 11.4.2. BidiRemoteSlave (BidiSlave) Filter

The BidiRemoteSlaveFilter is used by Tungsten Replicator to prevent statements that originated from this service (i.e. where data was extracted), being re-applied to the database. This is a requirement for replication to prevent data that may be transferred between hosts being re-applied, particularly in Active/Active and other bi-directional replication deployments.

| | |
|---|---|
| Pre-configured filter name | `bidiSlave` |
| Classname | `com.continuent.tungsten.replicator.filter.BidiRemoteSlaveFilter` |
| Property prefix | *`replicator.filter.bidiSlave`* |
| Stage compatibility | |
| tpm Option compatibility | |
| Data compatibility | Any event |
| Parameters | |

| Parameter | Type | Default | Description |
|---|---|---|---|
| *localServiceName* | string | *${local.service.name}* | Local service name of the service that reads the binary log |
| *allowBidiUnsafe* | boolean | false | If true, allows statements that may be unsafe for bi-directional replication |
| *allowAnyRemoteService* | boolean | false | If true, allows statements from any remote service, not just the current service |

The filter works by comparing the server ID of the THL event that was created when the data was extracted against the server ID of the current server.

When deploying through the tpm service the filter is automatically enabled for remote Appliers. For complex deployments, particularly those with bi-directional replication (including active/active), the *allowBidiUnsafe* parameter may need to be enabled to allow certain statements to be re-executed.

### Important

Known Issue for Active/Active installations with AWS Aurora

Due to a change in behaviour from MySQL v5.7 onwards, USER, VIEW, TRIGGER information is logged differently in the binary logs.

When AWS Aurora is in use as a Source, this change in behaviour prevents the filter from working correctly for DDL specific to USER (CREATE USER, GRANT etc), VIEWS (CREATE and DROP) and TRIGGERS (CREATE and DROP)

A current workaround would be to additionally use the dropDDL fiter until a future Tungsten Replicator release addresses the issue.

## 11.4.3. breadcrumbs.js Filter

The breadcrumbs filter records regular 'breadcrumb' points into a MySQL table for systems that do not have global transaction IDs. This can be useful if recovery needs to be made to a specific point. The example also shows how metadata information for a given event can be updated based on the information from a table.

| Pre-configured filter name | ansiquotes | | |
|---|---|---|---|
| JavaScript Filter File | tungsten-replicator/support/filters-javascript/breadcrumbs.js | | |
| Property prefix | *replicator.filter.breadcrumbs* | | |
| Stage compatibility | binlog-to-q | | |
| tpm Option compatibility | --svc-extractor-filters [425] | | |
| Data compatibility | Any event | | |
| Parameters | | | |
| Parameter | Type | Default | Description |
| *server_id* | numeric | (none) | MySQL server ID of the current host |

To use the filter:

1. A table is created and populated with one more rows on the Target server. For example:

```
CREATE TABLE `tungsten_svc1`.`breadcrumbs` (
 `id` int(11) NOT NULL PRIMARY KEY,
 `counter` int(11) DEFAULT NULL,
 `last_update` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP) ENGINE=InnoDB;
INSERT INTO tungsten_svc1.breadcrumbs(id, counter) values(@@server_id, 1);
```

2. Now set an event to update the table regularly. For example, within MySQL an event can be created for this purpose:

```
CREATE EVENT breadcrumbs_refresh
  ON SCHEDULE EVERY 5 SECOND
  DO
     UPDATE tungsten_svc1.breadcrumbs SET counter=counter+1;
SET GLOBAL event_scheduler = ON;
```

The filter will extract the value of the counter each time it sees to the table, and then mark each transaction with a particular server ID with the counter value plus an offset. For convenience we assume row replication is enabled.

If you need to failover to another server that has different logs, you can figure out the restart point by looking in the THL for the breadcrumb metadata on the last transaction. Use this to search the binary logs on the new server for the correct restart point.

The filter itself work in two stages, and operates because the JavaScript instance is persistent as long as the Replicator is running. This means that data extracted during replication stays in memory and can be applied to later transactions. Hence the breadcrumb ID and offset information can be identified and used on each call to the filter function.

The first part of the filter event identifies the breadcrumb table and extracts the identified breadcrumb counter:

```
if (table.compareToIgnoreCase("breadcrumbs") == 0)
{
  columnValues = oneRowChange.getColumnValues();
  for (row = 0; row < columnValues.size(); row++)
  {
    values = columnValues.get(row);
    server_id_value = values.get(0);
    if (server_id == null || server_id == server_id_value.getValue())
    {
      counter_value = values.get(1);
      breadcrumb_counter = counter_value.getValue();
      breadcrumb_offset = 0;
    }
  }
}
```

The second part updates the event metadata using the extracted breadcrumb information:

```
topLevelEvent = event.getDBMSEvent();
if (topLevelEvent != null)
{
  xact_server_id = topLevelEvent.getMetadataOptionValue("mysql_server_id");
  if (server_id == xact_server_id)
  {
    topLevelEvent.setMetaDataOption("breadcrumb_counter", breadcrumb_counter);
    topLevelEvent.setMetaDataOption("breadcrumb_offset", breadcrumb_offset);
  }
}
```

To calculate the offset (i.e. the number of events since the last breadcrumb value was extracted), the filter determines if the event was the last fragment processed, and updates the offset counter:

```
if (event.getLastFrag())
{
  breadcrumb_offset = breadcrumb_offset + 1;
}
```

## 11.4.4. CaseTransform Filter

The `CaseTransform` filter can be used to force convert Schema, Table and Column names to either upper or lower case.

| | | | |
|---|---|---|---|
| Pre-configured filter name | | `casetransform` | |
| Classname | | `com.continuent.tungsten.replicator.filter.CaseMappingFilter` | |
| Property prefix | | *`replicator.filter.casetransform`* | |
| Stage compatibility | | | |
| tpm Option compatibility | | | |
| Data compatibility | | Any Event | |
| Parameters | | | |
| Parameter | Type | Default | Description |
| *`to_upper_case`* | `boolean` | `true` | If true, converts object names to upper case; if false converts them to lower case |

This filter can be useful when replicating between environments that have different case sensitivity settings in place.

Uage Example

To force Upper Case on extractor:

```
svc-extractor-filters=casetransform
property=replicator.filter.casetransform.to_upper_case=true
```

To force Lower Case on applier:

```
svc-applier-filters=casetransform
property=replicator.filter.casetransform.to_upper_case=false
```

## 11.4.5. ColumnName Filter

The `ColumnNameFilter` loads the table specification information for tables and adds this information to the THL data for information extracted using row-base replication.

| | | | |
|---|---|---|---|
| Pre-configured filter name | | `colnames` | |
| Classname | | `com.continuent.tungsten.replicator.filter.ColumnNameFilter` | |
| Property prefix | | *`replicator.filter.colnames`* | |
| Stage compatibility | | `binlog-to-q` | |
| tpm Option compatibility | | `--svc-extractor-filters` [425] | |
| Data compatibility | | Row events | |
| Keeps Cached Data | | Yes | |
| Cached Refreshed When? | | Emptied when going `OFFLINE` [195]; Updated when `ALTER` statement seen | |
| Metadata Updated | | Yes; `tungsten_filter_columnname=true` | |
| Parameters | | | |
| Parameter | Type | Default | Description |
| *`user`* | string | *`${replicator.global.extract.db.user}`* | The username for the connection to the database for looking up column definitions |
| *`password`* | string | *`${replicator.global.extract.db.password}`* | The password for the connection to the database for looking up column definitions |
| *`url`* | string | `jdbc:mysql:thin://${replicator.global.extract.db.host}: » ${replicator.global.extract.db.port}/${replicator.schema}?createDB=true` | JDBC URL of the database connection to use for looking up column definitions |
| *`addSignedFlag`* | `boolean` | `true` | Determines whether the signed flag information for columns should be added to the metadata for each column. |
| *`ignoreMissingTables`* | `boolean` | `true` | When true, tables that do not exist will not trigger metadata and column names to be added to the THL data. |

### Note

This filter is designed to be used for testing and with heterogeneous replication where the field name information can be used to construct and build target data structures.

The filter is required for the correct operation of heterogeneous replication, for example when replicating to MongoDB. The filter works by using the replicator username and password to access the underlying database and obtain the table definitions. The table definition information is cached within the replication during operation to improve performance.

When extracting data from thew binary log using row-based replication, the column names for each row of changed data are added to the THL.

Enabling this filter changes the THL data from the following example, shown without the column names:

```
SEQ# = 27 / FRAG# = 0 (last frag)
- TIME = 2013-08-01 18:29:38.0
- EPOCH# = 11
- EVENTID = mysql-bin.000012:0000000000004369;0
- SOURCEID = host31
- METADATA = [mysql_server_id=1;dbms_type=mysql;service=alpha;shard=test]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1]
- SQL(0) =
 - ACTION = INSERT
 - SCHEMA = test
 - TABLE = sales
 - ROW# = 0
  - COL(1: ) = 1
  - COL(2: ) = 23
  - COL(3: ) = 45
```

```
  - COL(4: ) = 45000.00
```

To a version where the column names are included as part of the THL record:

```
SEQ# = 43 / FRAG# = 0 (last frag)
- TIME = 2013-08-01 18:34:18.0
- EPOCH# = 28
- EVENTID = mysql-bin.000012:0000000000006814;0
- SOURCEID = host31
- METADATA = [mysql_server_id=1;dbms_type=mysql;service=alpha;shard=test]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1]
- SQL(0) =
 - ACTION = INSERT
 - SCHEMA = test
 - TABLE = sales
 - ROW# = 0
  - COL(1: id) = 2
  - COL(2: country) = 23
  - COL(3: city) = 45
  - COL(4: value) = 45000.00
```

When the row-based data is applied to a non-MySQL database the column name information is used by the applier to specify the column, or they key when the column and value is used as a key/value pair in a document-based store.

## 11.4.6. ConvertStringFromMySQL Filter

The `ConvertStringFromMySQLFilter` is designed to be used in replicators that are used in conjunction either with existing native MySQL to MySQL replication deployments, or clustering deployments where the replication has been configured to use native MySQL byte storage for strings. These are incompatible with heterogeneous deployments as the string is stored internally and in the THL in a format that is useful only within similarly configured replicators.

Conversion can be selected to happen for all valid columns (`VARCHAR` or `CHAR` column types only), or for selected columns within specific tables and schemas. All conversions are made with the relevant character set for the table and THL event.

### Note

Conversion will not occur on incompatible columns. For example, conversion will not be applied to `INT` columns. This is the case even if the column has been explicitly set to convert the column.

| | | | |
|---|---|---|---|
| Pre-configured filter name | | *convertstringfrommysql* | |
| Classname | | `com.continuent.tungsten.replicator.ConvertStringFromMySQLFilter` | |
| Property prefix | | *Not defined* | |
| Stage compatibility | | any | |
| tpm Option compatibility | | | |
| Data compatibility | | Row events only | |
| Parameters | | | |
| *`definitionsFile`* | string | `support/filters-config/convertstringfrom-mysql.json` | JSON file containing the definition of which events and which tables to skip |

Configuration of the filter is made using the generic JSON file, which supports both default options to happen for all tables not otherwise explicitly specified. The default JSON file converts all valid (`VARCHAR` or `CHAR`) column types only:

```
{
    "__default": {
        "*" : "true",
    },
    "SCHEMA" : {
        "TABLE" : {
            "COLUMN" : "true",
        },
    }
}
```

### Warning

For column specific selection to work, the column names must be included within the THL. The `colnames` filter must have been enabled either before this filter, or on the extractor where the data was originally extracted.

The default section handles the default response when an explicit schema or table name does not appear. Further sections are then organised by schema, table and column name. Where the setting is `true`, conversion will take place. A `false` disables conversion.

To enable conversion on a single column `DESCRIPTION` within the SALES.INVOICE schema/table while disabling conversion on all other columns:

```
{
    "__default": {
        "*" : "false",
    },
    "SALES" : {
        "INVOICE" : {
            "DESCRIPTION" : "true",
        },
    }
}
```

To convert all compatible columns in all tables within a schema:

```
{
    "__default": {
        "*" : "false",
    },
    "SALES" : {
        "*" : {
            "*" : "true",
        },
    }
}
```

A primary use case for this filter is for Cluster-Extractor replication from a cluster to a datawarehouse. For more details, please see Replicating from a Cluster to a Datawarehouse.

Source Cluster Example

For Cluster-Extractor replication to a datawarehouse, the source cluster nodes must use ROW-based MySQL binary logging, and also must have two extractor filters enabled, `colnames` and `pkey`.

For example, on every cluster node the lines below would be added to the `/etc/tungsten/tungsten.ini` file in the service stanza, then tpm update would be executed:

```
repl-svc-extractor-filters=colnames,pkey
property=replicator.filter.pkey.addColumnsToDeletes=true
property=replicator.filter.pkey.addPkeyToInserts=true
```

For staging deployments, prepend two hyphens to each line and include on the command line.

For more details about configuring the source cluster, please see Section 3.4, “Replicating Data Out of a Cluster”.

Target Cluster-Extractor Example

On the replication Applier node, copy the `convertstringfrommysql.json` filter configuration sample file into the `/opt/continuent/share` directory then edit it to suit:

```
shell> cp /opt/continuent/tungsten/tungsten-replicator/support/filters-config/convertstringfrommysql.json /opt/continuent/share/
shell> vi /opt/continuent/share/convertstringfrommysql.json
```

Once the `convertstringfrommysql` JSON configuration file has been edited, update the `/etc/tungsten/tungsten.ini` file to add and configure the convertstringfrommysql filter.

For example, comfigure a service named `omega` on `host6` to read from the cluster nodes defined by cluster-alias `alpha`.

```
[alpha]
topology=cluster-alias
master=host1
members=host1,host2,host3
thl-port=2112

[omega]
topology=cluster-slave
relay=host6
relay-source=alpha
repl-svc-remote-filters=convertstringfrommysql
property=replicator.filter.convertstringfrommysql.definitionsFile=/opt/replicator/share/convertstringfrommysql.json
```

For more details about configuring the target Cluster-Extractor node, please see Section 3.4, “Replicating Data Out of a Cluster”.

## 11.4.7. DatabaseTransform (dbtransform) Filter

This filter can be used to rename databases (schemas) and/or tables between source and targets

| | | | |
|---|---|---|---|
| Pre-configured filter name | | dbtransform | |
| Classname | | com.continuent.tungsten.replicator.filter.DatabaseTransformFilter | |
| Property prefix | | *replicator.filter.dbtransform* | |
| Stage compatibility | | | |
| tpm Option compatibility | | | |
| Data compatibility | | ROW Events only | |
| Parameters | | | |
| Parameter | Type | Default | Description |
| *transformTables* | boolean | false | If set to true, forces the rename transformations to operate on tables, not databases |
| *from_regex1* | string | foo | The search regular expression to use when renaming databases or tables (group 1); corresponds to *to_regex1* |
| *to_regex1* | string | bar | The replace regular expression to use when renaming databases or tables (group 1); corresponds to *from_regex1* |
| *from_regex2* | string | | The search regular expression to use when renaming databases or tables (group 2); corresponds to *to_regex2* |
| *to_regex2* | string | | The replace regular expression to use when renaming databases or tables (group 2); corresponds to *from_regex2* |
| *from_regex3* | string | | The search regular expression to use when renaming databases or tables (group 3); corresponds to *to_regex3* |
| *to_regex3* | string | | The replace regular expression to use when renaming databases or tables (group 3); corresponds to *from_regex3* |
| *from_regex4* | string | | The search regular expression to use when renaming databases or tables (group 4); corresponds to *to_regex4* |
| *to_regex4* | string | | The replace regular expression to use when renaming databases or tables (group 4); corresponds to *from_regex4* |

The dbtransoform filter can be used to apply standard Java Regex expressions to rename databases and/or tables between source and target.

Up to 4 from/to regex patterns can be provided. By default the transofrmation will be applied to Database Schema names. To use the transofrm for Table Names, specify the transformTables=true option.

The filter will only transform database or tables, not a mix of the two. For more advanced transofrmation you may want to consider the rename filter instead

The filter only wors with ROW events. Statement based transofrmation are not supported with this filter.

## 11.4.8. dbrename.js Filter

The dbrename JavaScript filter renames database (schemas) using two parameters from the properties file, the *dbsource* and *dbtarget*. Each event is then processed, and the statement or row based schema information is updated to *dbtarget* when the *dbsource* schema is identified.

| | | | |
|---|---|---|---|
| Pre-configured filter name | | dbrename | |
| JavaScript Filter File | | tungsten-replicator/support/filters-javascript/dbrename.js | |
| Property prefix | | *replicator.filter.dbrename* | |
| Stage compatibility | | binlog-to-q | |
| tpm Option compatibility | | --svc-extractor-filters [425] | |
| Data compatibility | | Any event | |
| Parameters | | | |
| Parameter | Type | Default | Description |
| *dbsource* | string | (none) | Source table name (database/table to be renamed) |

| | | | |
|---|---|---|---|
| *dbtarget* | string | [none] | New database/table name |

To configure the filter you would add the following to your properties:

```
replicator.filter.dbrename=com.continuent.tungsten.replicator.filter.JavaScriptFilter
replicator.filter.dbrename.script=${replicator.home.dir}/samples/extensions/javascript/dbrename.js
replicator.filter.dbrename.dbsource=SOURCE
replicator.filter.dbrename.dbtarget=TEST
```

The operation of the filter is straightforward, because the schema name is exposed and settable within the statement and row change objects:

```
function filter(event)
{
    sourceName = filterProperties.getString("dbsource");
    targetName = filterProperties.getString("dbtarget");

    data = event.getData();

    for(i=0;i<data.size();i++)
    {
        d = data.get(i);

        if(d instanceof
           com.continuent.tungsten.replicator.dbms.StatementData)
        {
            if(d.getDefaultSchema() != null &&
               d.getDefaultSchema().compareTo(sourceName)==0)
            {
                d.setDefaultSchema(targetName);
            }
        }
        else if(d instanceof
                com.continuent.tungsten.replicator.dbms.RowChangeData)
        {
            rowChanges = data.get(i).getRowChanges();

            for(j=0;j<rowChanges.size();j++)
            {
                oneRowChange = rowChanges.get(j);

                if(oneRowChange.getSchemaName().compareTo(sourceName)==0)
                {
                    oneRowChange.setSchemaName(targetName);
                }
            }
        }
    }
}
```

## 11.4.9. dbselector.js Filter

Filtering only a single database schema can be useful when you want to extract a single schema for external processing, or for sharding information across multiple replication targets. The `dbselector` filter deletes all statement and row changes, except those for the selected table. To configure, the *db* parameter to the filter configuration specifies the schema to be replicated.

| | | | |
|---|---|---|---|
| Pre-configured filter name | | dbselector | |
| JavaScript Filter File | | tungsten-replicator/support/filters-javascript/dbselector.js | |
| Property prefix | | *replicator.filter.dbselector* | |
| Stage compatibility | | binlog-to-q, q-to-thl, q-to-dbms | |
| tpm Option compatibility | | --svc-extractor-filters [425], --svc-applier-filters [424] | |
| Data compatibility | | Any event | |
| Parameters | | | |
| Parameter | Type | Default | Description |
| *db* | string | [none] | Database to be selected |

Within the filter, statement changes look for the schema in the `StatementData` object and remove it from the array:

```
if (d instanceof com.continuent.tungsten.replicator.dbms.StatementData)
{
    if(d.getDefaultSchema().compareTo(db)!=0)
    {
```

```
        data.remove(i);
        i--;
    }
}
```

Because entries are being removed from the list of statements, the iterator used to process each item must be explicitly decremented by 1 to reset the counter back to the new position.

Similarly, when looking at row changes in the `RowChangeData`:

```
else if(d instanceof com.continuent.tungsten.replicator.dbms.RowChangeData)
{
    rowChanges = data.get(i).getRowChanges();

    for(j=0;j<rowChanges.size();j++)
    {
        oneRowChange = rowChanges.get(j);

        if(oneRowChange.getSchemaName().compareTo(db)!=0)
        {
            rowChanges.remove(j);
            j--;
        }
    }
}
```

## 11.4.10. `dbupper.js` Filter

The `dbupper` filter changes the case of the schema name for all schemas to uppercase. The schema information is easily identified in the statement and row based information, and therefore easy to update.

| | | | |
|---|---|---|---|
| Pre-configured filter name | | `dbupper` | |
| JavaScript Filter File | | `tungsten-replicator/support/filters-javascript/dbupper.js` | |
| Property prefix | | *`replicator.filter.dbupper`* | |
| Stage compatibility | | `binlog-to-q` | |
| tpm Option compatibility | | `--svc-extractor-filters` [425], `--svc-applier-filters` [424] | |
| Data compatibility | | Any event | |
| Parameters | | | |
| Parameter | Type | Default | Description |
| *`from`* | `string` | (none) | Database name to be converted to uppercase |

For example, within statement data:

```
from = d.getDefaultSchema();
if (from != null)
{
    to   = from.toUpperCase();
    d.setDefaultSchema(to);
}
```

## 11.4.11. `dropcolumn.js` Filter

The `dropcolumn` filter enables columns in the THL to be dropped. This can be useful when replicating Personal Identification Information, such as email addresses, phone number, personal identification numbers and others are within the THL but need to be filtered out on the Target.

| | | | |
|---|---|---|---|
| Pre-configured filter name | | `dropcolumn` | |
| JavaScript Filter File | | `tungsten-replicator/support/filters-javascript/dropcolumn.js` | |
| Property prefix | | *`replicator.filter.dropcolumn`* | |
| Stage compatibility | | `binlog-to-q`, `q-to-dbms` | |
| tpm Option compatibility | | `--svc-extractor-filters` [425], `--svc-applier-filters` [424] | |
| Data compatibility | | Any event | |
| Parameters | | | |
| Parameter | Type | Default | Description |
| *`definitionsFile`* | Filename | `~/dropcolumn.json` | Location of the definitions file for dropping columns |

| | | | |
|---|---|---|---|
| *fillGaps* | Boolean | false | When true, re-order THL Column Index IDs to sequential numbers |

The filter is available by default as dropcolumn, and the filter is configured through a JSON file that defines the list of columns to be dropped. The filter relies on the colnames filter being enabled.

To enable the filter:

```
shell> tpm update --svc-extractor-filters=colnames,dropcolumn \
    --property=replicator.filter.dropcolumn.definitionsFile=/opt/continuent/share/dropcolumn.json
```

A sample configuration file is provided in tungsten-replicator/support/filters-config/dropcolumn.json. The format of the file is a JSON array of schema/table/column specifications:

```
[
    {
        "schema": "vip",
        "table": "clients",
        "columns": [
            "personal_code",
            "birth_date",
            "email"
        ]
    },
    ...
]
```

Where:

- schema specifies the name of the schema on which to apply the filtering. If * is given, all schemas are matched.
- table specifies the name of the table on which to apply the filtering. If * is given, all tables are matched.
- columns is an array of column names to be matched.

For example:

```
[
    {
        "schema": "vip",
        "table": "clients",
        "columns": [
            "personal_code",
            "birth_date",
            "email"
        ]
    },
    ...
]
```

Filters the columns email, birth_date, and personal_code within the clients table in the vip schema.

To filter the telephone column in any table and any schema:

```
[
    {
        "schema": "*",
        "table": "*",
        "columns": [
            "telephone"
        ]
    }
]
```

Care should be taken when dropping columns on the Target and Source when the column order is different or when the names of the column differ:

- If the column order is same, even if dropcolumn.js is used, leave the default setting for the property *replicator.applier.dbms.getColumnMetadataFromDB=true*.
- If the column order is different on the Source and Target, set *replicator.applier.dbms.getColumnMetadataFromDB=false*
- If slave's column names are different, regardless of differences in the order, use the default property setting *replicator.applier.dbms.getColumnMetadataFromDB=true*

If the filter is enabled on the extractor, the columns will be removed from the THL and the resulting THL Index will appear to have gaps in the THL column index ID, for example:

```
...
- SQL(0) =
 - ACTION = INSERT
 - SCHEMA = sample
 - TABLE = demotable
 - ROW# = 0
  - COL(1: id) = 11
  - COL(2: col_a) = UK
  - COL(4: col_c) = 5678
  - COL(5: col_d) = ABC
  - COL(7: col_g) = 2019-09-05 07:21:48.0
  - KEY(1: id) = NULL
...
```

For JDBC targets, this is expected and required to ensure accurate column mapping, however for Batch Appliers the gap in the Index ID's will cause the applier to fail with a CSV Column Mismatch error, therefore, if your target is a Batch target (eg Hadoop, Redshift, Vertica) you need to add the following property to your configuration:

```
property=replicator.filter.dropcolumn.fillGaps=true
```

With this property in place, the resulting THL from the above example would now look like the following:

```
...
- SQL(0) =
 - ACTION = INSERT
 - SCHEMA = sample
 - TABLE = demotable
 - ROW# = 0
  - COL(1: id) = 11
  - COL(2: col_a) = UK
  - COL(3: col_c) = 5678
  - COL(4: col_d) = ABC
  - COL(5: col_g) = 2019-09-05 07:21:48.0
  - KEY(1: id) = NULL
...
```

## 11.4.12. `dropcomments.js` Filter

The `dropcomments` filter removes comments from statements within the event data.

| | | | |
|---|---|---|---|
| Pre-configured filter name | | `dropcomments` | |
| JavaScript Filter File | | `tungsten-replicator/support/filters-javascript/dropcomments.js` | |
| Property prefix | | *`replicator.filter.dropcomments`* | |
| Stage compatibility | | `binlog-to-q`, `q-to-dbms` | |
| tpm Option compatibility | | `--svc-extractor-filters` [425], `--svc-applier-filters` [424] | |
| Data compatibility | | Any event | |
| Parameters | | | |
| Parameter | Type | Default | Description |

Row changes do not have comments, so the filter only has to change the statement information, which is achieved by using a regular expression:

```
sqlOriginal = d.getQuery();
sqlNew = sqlOriginal.replaceAll("/\\*(?:.|[\\n\\r])*?\\*/","");
d.setQuery(sqlNew);
```

To handle the case where the statement could only be a comment, the statement is removed:

```
if(sqlNew.trim().length()==0)
{
    data.remove(i);
    i--;
}
```

## 11.4.13. `dropddl.js` Filter

### Note

Note that this filter is only intended for use with MySQL<>MySQL replication and will not have any benefit at this time for heterogeneous replication.

There may be occasions where you do not require specific DDL statements to be replicated. For example, you may NOT want to allow TRIGGERS or VIEWS to be replicated, but you do want TABLES, INDEXES, FUNCTIONS and PROCEDURES. The dropddl filter is intended for this purpose.

The dropddl filter allows you to configure which, if any, ddl statements to drop from THL and not be replicated.

| | | | |
|---|---|---|---|
| Pre-configured filter name | | `dropddl` | |
| JavaScript Filter File | | `tungsten-replicator/support/filters-javascript/dropddl.js` | |
| Property prefix | | *`replicator.filter.dropddl`* | |
| Stage compatibility | | `binlog-to-q`, `q-to-dbms` | |
| tpm Option compatibility | | `--svc-extractor-filters` [425], `--svc-applier-filters` [424] | |
| Data compatibility | | Any event | |
| Parameters | | | |
| Parameter | Type | Default | Description |
| dropTableDDL | boolean | false | Drop CREATE TABLE and DROP TABLE statements |
| dropViewDDL | boolean | false | Drop CREATE VIEW and DROP VIEWS statements |
| dropIndexDDL | boolean | false | Drop CREATE INDEX and DROP INDEX statements |
| dropTriggerDDL | boolean | false | Drop CREATE TRIGGER and DROP TRIGGER statements |
| dropFunctionDDL | boolean | false | Drop CREATE FUNCTION and DROP FUNCTION statements |
| dropProcedureDDL | boolean | false | Drop CREATE PROCEDURE and DROP PROCEDURE statements |
| dropUserDDL | boolean | false | Drop CREATE USER and ALTER USER statements |
| dropGrantDDL | boolean | false | Drop GRANT and REVOKE statements |

Example:

If you wish to drop VIEW and TRIGGER DDL on the applier, the following options need to be added to your configuration:

```
svc-applier-filters=dropddl
property=replicator.filter.dropddl.dropViewDDL=true
property=replicator.filter.dropddl.dropTriggerDDL=true
```

## 11.4.14. `dropmetadata.js` Filter

All events within the replication stream contain metadata about each event. This information can be individually processed and manipulated. The `dropmetadata` filter removes specific metadata from each event, configured through the *`option`* parameter to the filter.

| | | | |
|---|---|---|---|
| Pre-configured filter name | | `dropmetadata` | |
| JavaScript Filter File | | `tungsten-replicator/support/filters-javascript/dropmetadata.js` | |
| Property prefix | | *`replicator.filter.ansiquotes`* | |
| Stage compatibility | | `binlog-to-q`, `q-to-dbms` | |
| tpm Option compatibility | | `--svc-extractor-filters` [425], `--svc-applier-filters` [424] | |
| Data compatibility | | Any event | |
| Parameters | | | |
| Parameter | Type | Default | Description |
| *`option`* | `string` | [none] | Name of the metadata field to be dropped |

Metadata information can be processed at the event top-level:

```
metaData = event.getDBMSEvent().getMetadata();
for(m = 0; m < metaData.size(); m++)
{
    option = metaData.get(m);
    if(option.getOptionName().compareTo(optionName)==0)
    {
        metaData.remove(m);
        break;
```

```
    }
}
```

## 11.4.15. droprow.js Filter

The droprow filter can be used to selectively filter out transactions based on matching column values at the ROW level.

| Pre-configured filter name | | droprow | |
|---|---|---|---|
| JavaScript Filter File | | tungsten-replicator/support/filters-javascript/droprow.js | |
| Property prefix | | *replicator.filter.droprow* | |
| Stage compatibility | | binlog-to-q, q-to-dbms | |
| tpm Option compatibility | | --svc-extractor-filters [425], --svc-applier-filters [424] | |
| Data compatibility | | Row Events | |
| Parameters | | | |
| Parameter | Type | Default | Description |
| *definitionsFile* | Filename | ~/droprow.json | Location of the definitions file for row filtering. |
| *matchCase* | Boolean | true | Controls whether text based comparisons are Case Sensitive (true) or not (false) |
| *rule* | String | matchany | Valid Values : matchany \| matchall. If more than one column/value pair are defined, this property will determine whether there must be a match for all columns (matchall) or only one (matchany). |

The filter is available by default as droprow, and the filter is configured through a JSON file that defines the list of column/values to match in a row to be dropped.

This filter has the following requirements and caveats:

- The filter relies on the colnames filter being enabled.
- The filter will only work on ROW events, therefore the source database needs to be running in Row-Based binary logging mode.
- The filter will only compare values as part of an INSERT satement and NEW values as part of an UPDATE statement.
- Rows that are affected by UPDATE and DELETE statements where the values are part of the WHERE clause will not be removed from THL.
- If the filter is applied to tables with Foreign Keys, be sure to inclue all tables in the hierarchy as part of the filter to avoid referential integrity errors.

To enable the filter, for staging based deployments:

```
shell> tpm update --svc-extractor-filters=colnames,droprow \
    --property=replicator.filter.droprow.definitionsFile=/opt/continuent/share/droprow.json
```

Additional parameters that override the defaults can be also supplied, for example:

```
shell> tpm update --svc-extractor-filters=colnames,droprow \
    --property=replicator.filter.droprow.definitionsFile=/opt/continuent/share/droprow.json \
    --property=replicator.filter.droprow.rule=matchall
```

To enable the filter, for ini based deployments:

```
shell> vi /etc/tungsten/tungsten.ini

[servicename]
...
svc-extractor-filters=colnames,droprow
property=replicator.filter.droprow.definitionsFile=/opt/continuent/share/droprow.json
...

shell> tpm update
```

Additional parameters that override the defaults can be also supplied, for example:

```
shell> vi /etc/tungsten/tungsten.ini
```

```
[servicename]
...
svc-extractor-filters=colnames,droprow
property=replicator.filter.droprow.definitionsFile=/opt/continuent/share/droprow.json
property=replicator.filter.droprow.rule=matchall
...

shell> tpm update
```

A sample configuration file is provided in `tungsten-replicator/support/filters-config/droprow.json`. The format of the file is a JSON array of schema/table/column/value specifications. The match rule can also be supplied on a per table basis as per the examples below. If not supplied the global default for the filter, as described above, will be used:

```
[
  {
    "schema": "vip",
    "table": "clients",
    "rule": "matchall",
    "columns": [
      {
        "column": "city",
        "value": "London"
      },
      {
        "column": "country",
        "value": "UK"
      }
    ]
  }
]
```

Where:

- `schema` specifies the name of the schema on which to apply the filtering.
- `table` specifies the name of the table on which to apply the filtering.
- `rule` specifies the matching rule to apply the filtering. Setting the value in the JSON will override the global default specified in the main configuration (See above).

  `matchany`(default) will remove rows if only one condition matches. Equivalent to an OR condition.

  `matchall` will remove rows if only all conditions match. Equivalent to an AND condition.
- `columns` is an array of column/values to be matched.
- `column` is the name of the column in the table to match.
- `value` is the Value of the column to match. When Rule is `matchany` this can also be supplied as an array of Values (See second example below).

For example:

```
[
  {
    "schema": "vip",
    "table": "customers",
    "rule": "matchany",
    "columns": [
      {
        "column": "country",
        "value": [ "Australia", "UK" ]
      }
    ]
  }
]
```

The bove example filters rows from the customers table, in the vip schema where the country column is either "Australia" OR "UK"

## 11.4.16. `dropstatementdata.js` Filter

Within certain replication deployments, enforcing that only row-based information is replicated is important to ensure that the row data is replicated properly. For example, when replicating to databases that do not accept statements, these events must be filtered out.

| | |
|---|---|
| Pre-configured filter name | `dropstatementdata` |

| | | | |
|---|---|---|---|
| JavaScript Filter File | | tungsten-replicator/support/filters-javascript/dropstatementdata.js | |
| Property prefix | | *replicator.filter.dropstatementdata* | |
| Stage compatibility | | binlog-to-q, q-to-dbms | |
| tpm Option compatibility | | --svc-extractor-filters [425], --svc-applier-filters [424] | |
| Data compatibility | | Any event | |
| Parameters | | | |
| Parameter | Type | Default | Description |

This is achieved by checking for statements, and then removing them from the event:

```
data = event.getData();

for(i = 0; i < data.size(); i++)
{
    d = data.get(i);

    if(d instanceof com.continuent.tungsten.replicator.dbms.StatementData)
    {
        data.remove(i);
        i--;
    }
}
```

## 11.4.17. dropsqlmode.js Filter

Different releases of MySQL occasionally add/remove various sql_modes, and in doing so if replicating between different versions, you may receive errors during replication as MySQL will reject unknown sql_modes.

This filter was specifically added to handle replication between MySQL 5.7 and MySQL 8 where a number of sql_modes where removed.

| | | | |
|---|---|---|---|
| Pre-configured filter name | | dropsqlmode | |
| JavaScript Filter File | | tungsten-replicator/support/filters-javascript/dropsqlmode.js | |
| Property prefix | | *replicator.filter.dropsqlmode* | |
| Stage compatibility | | q-to-dbms | |
| tpm Option compatibility | | --svc-applier-filters [424] | |
| Data compatibility | | Any event | |
| Parameters | | | |
| Parameter | Type | Default | Description |
| modes | string | See below | \| separated string of sql_modes to drop |

By default, the filter is configured to drop the following sql_modes: NO_AUTO_CREATE_USER,NO_FIELD_OPTIONS,NO_KEY_OPTIONS,NO_TABLE_OPTIONS

If you wish to add/remove sql_modes from this list, you will need to override the property as follows:

```
property=replicator.filter.dropsqlmode.modes=NO_AUTO_CREATE_USER|NO_FIELD_OPTIONS|NO_KEY_OPTIONS|NO_TABLE_OPTIONS| TIME_TRUNCATE_FRACTIONAL
```

## 11.4.18. dropxa.js Filter

The dropxa filter removes XA Transaction Events within the event data.

| | | | |
|---|---|---|---|
| Pre-configured filter name | | dropxa | |
| JavaScript Filter File | | tungsten-replicator/support/filters-javascript/dropxa.js | |
| Property prefix | | *replicator.filter.dropxa* | |
| Stage compatibility | | binlog-to-q | |
| tpm Option compatibility | | --svc-extractor-filters [425] | |
| Data compatibility | | Any event | |
| Parameters | | | |
| Parameter | Type | Default | Description |

XA Transaction support was added into the replicator in version 6.1.14 however this filter can still be used if required to remove XA Transaction events if they are not required for replication

## 11.4.19. Dummy Filter

| | |
|---|---|
| Pre-configured filter name | `dummy` |
| Classname | `com.continuent.tungsten.replicator.filter.DummyFilter` |
| Property prefix | *`replicator.filter.dummy`* |
| Stage compatibility | |
| tpm Option compatibility | |
| Data compatibility | Any event |
| Parameters | |
| (none) | |

As the name suggests, the `dummy` filter does nothing, however it can be used as a simple mechanism to ensure that filters load correctly.

## 11.4.20. EnumToString Filter

The `EnumToString` filter translates `ENUM` datatypes within MySQL tables into their string equivalent within the THL.

| | | | |
|---|---|---|---|
| Pre-configured filter name | | `enumtostring` | |
| Classname | | `com.continuent.tungsten.replicator.filter.EnumToStringFilter` | |
| Property prefix | | *`replicator.filter.enumtostring`* | |
| Stage compatibility | | `binlog-to-q` | |
| tpm Option compatibility | | `--repl-svc-extractor-filters` [425] | |
| Data compatibility | | Row events | |
| Metadata Updated | | Yes; `tungsten_filter_enumtostring=true` | |
| Parameters | | | |
| Parameter | Type | Default | Description |
| *`user`* | string | *`${replicator.global.extract.db.user}`* | The username for the connection to the database for looking up column definitions |
| *`password`* | string | *`${replicator.global.extract.db.password}`* | The password for the connection to the database for looking up column definitions |
| *`url`* | string | `jdbc:mysql:thin://${replicator.global.extract.db.host}: » ${replicator.global.extract.db.port}/${replicator.schema}?createDB=true` | JDBC URL of the database connection to use for looking up column definitions |
| *`reconnectTimeout`* | integer | 3600 | Timeout for refreshing connection to database |

The `EnumToString` filter should be used with heterogeneous replication to ensure that the data is represented as the string value, not the internal numerical representation.

In the THL output below, the table has a `ENUM` column, `country`:

```
mysql> describe salesadv;
+----------+-----------------------------------+------+-----+---------+----------------+
| Field    | Type                              | Null | Key | Default | Extra          |
+----------+-----------------------------------+------+-----+---------+----------------+
| id       | int(11)                           | NO   | PRI | NULL    | auto_increment |
| country  | enum('US','UK','France','Australia') | YES  |     | NULL    |                |
| city     | int(11)                           | YES  |     | NULL    |                |
| salesman | set('Alan','Zachary')             | YES  |     | NULL    |                |
| value    | decimal(10,2)                     | YES  |     | NULL    |                |
+----------+-----------------------------------+------+-----+---------+----------------+
```

When extracted in the THL, the representation uses the internal value (for example, 1 for the first enumerated value). This can be seen in the THL output below.

```
SEQ# = 138 / FRAG# = 0 (last frag)
- TIME = 2013-08-01 19:09:35.0
```

```
- EPOCH# = 122
- EVENTID = mysql-bin.000012:0000000000021434;0
- SOURCEID = host31
- METADATA = [mysql_server_id=1;dbms_type=mysql;service=alpha;shard=test]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1]
- SQL(0) =
 - ACTION = INSERT
 - SCHEMA = test
 - TABLE = salesadv
 - ROW# = 0
  - COL(1: id) = 2
  - COL(2: country) = 1
  - COL(3: city) = 8374
  - COL(4: salesman) = 1
  - COL(5: value) = 35000.00
```

For the `country` column, the corresponding value in the THL is `1`. With the `EnumToString` filter enabled, the value is expanded to the corresponding string value:

```
SEQ# = 121 / FRAG# = 0 (last frag)
- TIME = 2013-08-01 19:05:14.0
- EPOCH# = 102
- EVENTID = mysql-bin.000012:0000000000018866;0
- SOURCEID = host31
- METADATA = [mysql_server_id=1;dbms_type=mysql;service=alpha;shard=test]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1]
- SQL(0) =
 - ACTION = INSERT
 - SCHEMA = test
 - TABLE = salesadv
 - ROW# = 0
  - COL(1: id) = 1
  - COL(2: country) = US
  - COL(3: city) = 8374
  - COL(4: salesman) = Alan
  - COL(5: value) = 35000.00
```

The information is critical when applying the data to a dataserver that is not aware of the table definition when replicating to non-MySQL target.

The examples here also show the Section 11.4.38, “SetToString Filter” and Section 11.4.5, “ColumnName Filter” filters.

## 11.4.21. EventMetadata Filter

Filters events based on metadata; used by default within sharding and Active/Active topologies

| | |
|---|---|
| Pre-configured filter name | `eventmetadata` |
| Classname | `com.continuent.tungsten.replicator.filter.EventMetadataFilter` |
| Property prefix | *`replicator.filter.eventmetadata`* |
| Stage compatibility | |
| tpm Option compatibility | |
| Data compatibility | Row events |
| Metadata Updated | Yes; `tungsten_filter_settostring=true` |
| Parameters | |
| (none) | |

## 11.4.22. `foreignkeychecks.js` Filter

The `foreignkeychecks` filter switches off foreign key checks for statements using the following statements:

```
CREATE TABLE
DROP TABLE
ALTER TABLE
RENAME TABLE
```

| | |
|---|---|
| Pre-configured filter name | `foreignkeychecks` |
| JavaScript Filter File | `tungsten-replicator/support/filters-javascript/foreignkeychecks.js` |

| | | | |
|---|---|---|---|
| Property prefix | | *replicator.filter.foreignkeychecks* | |
| Stage compatibility | | binlog-to-q, q-to-dbms | |
| tpm Option compatibility | | --svc-extractor-filters [425], --svc-applier-filters [424] | |
| Data compatibility | | Any event | |
| Parameters | | | |
| Parameter | Type | Default | Description |

The process checks the statement data and parses the content of the SQL statement by first trimming any extraneous space, and then converting the statement to upper case:

```
upCaseQuery = d.getQuery().trim().toUpperCase();
```

Then comparing the string for the corresponding statement types:

```
if(upCaseQuery.startsWith("CREATE TABLE") ||
    upCaseQuery.startsWith("DROP TABLE") ||
    upCaseQuery.startsWith("ALTER TABLE") ||
    upCaseQuery.startsWith("RENAME TABLE")
)
{
```

If they match, a new statement is inserted into the event that disables foreign key checks:

```
query = "SET foreign_key_checks=0";
newStatement = new com.continuent.tungsten.replicator.dbms.StatementData(
     d.getDefaultSchema(),
     null,
     query
     );
data.add(0, newStatement);
i++;
```

The use of `0` in the `add()` method inserts the new statement before the others within the current event.

## 11.4.23. Heartbeat Filter

The heartbeat file detects heartbeat events on Sources or Targets

| | | | |
|---|---|---|---|
| Pre-configured filter name | | (none) | |
| Classname | | com.continuent.tungsten.replicator.filter.HeartbeatFilter | |
| Property prefix | | (none) | |
| Stage compatibility | | | |
| tpm Option compatibility | | | |
| Data compatibility | | Any event | |
| Parameters | | | |
| Parameter | Type | Default | Description |
| *heartbeatInterval* | numeric | 3000 | Interval in milliseconds when a heartbeat event is inserted into the THL |

## 11.4.24. insertsonly.js Filter

The `insertsonly` filter filters events to only include ROW-based events using `INSERT`.

| | | | |
|---|---|---|---|
| Pre-configured filter name | | insertsonly | |
| JavaScript Filter File | | tungsten-replicator/support/filters-javascript/insertonly.js | |
| Property prefix | | *replicator.filter.insertonly* | |
| Stage compatibility | | q-to-dbms | |
| tpm Option compatibility | | --svc-applier-filters [424] | |
| Data compatibility | | Row events | |
| Parameters | | | |

| Parameter | Type | Default | Description |
|---|---|---|---|

This is achieved by examining each row and removing row changes that do not match the `INSERT` action type:

```
if(oneRowChange.getAction()!="INSERT")
{
    rowChanges.remove(j);
    j--;
}
```

## 11.4.25. Logging Filter

Logs filtered events through the standard replicator logging mechanism

| | |
|---|---|
| Pre-configured filter name | `logger` |
| Classname | `com.continuent.tungsten.replicator.filter.LoggingFilter` |
| Property prefix | *`replicator.filter.logger`* |
| Stage compatibility | |
| tpm Option compatibility | |
| Data compatibility | Any event |
| Parameters | |
| (none) | |

## 11.4.26. MySQLSessionSupport (mysqlsessions) Filter

Filters transactions for session specific temporary tables and variables

| | |
|---|---|
| Pre-configured filter name | `mysqlsessions` |
| Classname | `com.continuent.tungsten.replicator.filter.MySQLSessionSupportFilter` |
| Property prefix | *`replicator.filter.mysqlsession`* |
| Stage compatibility | |
| tpm Option compatibility | |
| Data compatibility | Any event |
| Parameters | |
| (none) | |

## 11.4.27. mapcharset Filter

### Note

This filter was introduced in version 6.1.12

Maps charsets in newer MySQL 8 environments allowing THL to be applied to older versions of MySQL.

| | |
|---|---|
| Pre-configured filter name | `mapcharset` |
| Classname | `com.continuent.tungsten.replicator.filter.mapcharset` |
| Property prefix | *`replicator.filter.mapcharset`* |
| Stage compatibility | q-to-dbms |
| tpm Option compatibility | --svc-applier-filters |
| Data compatibility | Any event |
| Parameters | |
| (none) | |

### Warning

The filter must not be used if applying into MySQL 8 or greater.

Once a replica has been upgraded to MySQL 8 or greater, the filter MUST be disabled.

The filter utilisies the `tungsten-replicator/support/filters-config/mapcharset.json` to determine the source/target charset mappings.

The json file is a very simple array structure in the format of <SourceCharset>: <TargetCharset>

The default installation of this filter should cover most, if not all, scenarios.

This, however, can be changed by updating `mapcharset.json` file which maps MySQL collation id, as seen by the following mysql query example:

```
mysql> select * from INFORMATION_SCHEMA.COLLATIONS where character_set_name like 'utf8mb4' order by id;

+----------------------------+--------------------+-----+------------+-------------+---------+---------------+
| COLLATION_NAME             | CHARACTER_SET_NAME | ID  | IS_DEFAULT | IS_COMPILED | SORTLEN | PAD_ATTRIBUTE |
+----------------------------+--------------------+-----+------------+-------------+---------+---------------+
| utf8mb4_general_ci         | utf8mb4            |  45 |            | Yes         |       1 | PAD SPACE     |
| utf8mb4_bin                | utf8mb4            |  46 |            | Yes         |       1 | PAD SPACE     |
| utf8mb4_unicode_ci         | utf8mb4            | 224 |            | Yes         |       8 | PAD SPACE     |
| utf8mb4_icelandic_ci       | utf8mb4            | 225 |            | Yes         |       8 | PAD SPACE     |
...
| utf8mb4_0900_ai_ci         | utf8mb4            | 255 | Yes        | Yes         |       0 | NO PAD        |
| utf8mb4_de_pb_0900_ai_ci   | utf8mb4            | 256 |            | Yes         |       0 | NO PAD        |
| utf8mb4_is_0900_ai_ci      | utf8mb4            | 257 |            | Yes         |       0 | NO PAD        |
| utf8mb4_lv_0900_ai_ci      | utf8mb4            | 258 |            | Yes         |       0 | NO PAD        |
| utf8mb4_ro_0900_ai_ci      | utf8mb4            | 259 |            | Yes         |       0 | NO PAD        |
...
| utf8mb4_ru_0900_as_cs      | utf8mb4            | 307 |            | Yes         |       0 | NO PAD        |
| utf8mb4_zh_0900_as_cs      | utf8mb4            | 308 |            | Yes         |       0 | NO PAD        |
| utf8mb4_0900_bin           | utf8mb4            | 309 |            | Yes         |       1 | NO PAD        |
+----------------------------+--------------------+-----+------------+-------------+---------+---------------+
75 rows in set (0.00 sec)
```

For example, the default mapping contains :

```
{
...
  "255": "45"
...
}
```

This means that collation `utf8mb4_0900_ai_ci` is mapped to `utf8mb4_general_ci`.

`tungsten-replicator/support/filters-config/mapcharset.readme` contains a full list of the default mappings.

## 11.4.28. NetworkClient Filter

The `NetworkClientFilter` processes data in selected columns

| | |
|---|---|
| Pre-configured filter name | `networkclient` |
| Classname | `com.continuent.tungsten.replicator.filter.NetworkClientFilter` |
| Property prefix | *`replicator.filter.networkclient`* |
| Stage compatibility | Any |
| tpm Option compatibility | `--svc-extractor-filters` [425], `--svc-thl-filters` [426], `--svc-applier-filters` [424] |
| Data compatibility | Row events |

Parameters

| Parameter | Type | Default | Description |
|---|---|---|---|
| *`definitionsFile`* | pathname | ${replicator.home.dir}/samples/extensions/java/network-client.json | The name of a file containing the definitions for how columns should be processed by filters |
| *`serverPort`* | number | 3112 | The network port to use when communicating with the network client |
| *`timeout`* | number | 10 | Timeout in seconds before treating the network client as failed when waiting to send or receive content. |

The network filter operates by sending field data, as defined in the corresponding filter configuration file, out to a network server that processes the information and sends it back to be re-introduced in place of the original field data. This can be used to translate and reformat information during the replication scheme.

The filter operation works as follows:

- All filtered data will be sent to a single network server, at the configured port.
- A single network server can be used to provide multiple transformations.
- The JSON configuration file for the filter supports multiple types and multiple column definitions.
- The protocol used by the network filter must be followed to effectively process the information. A failure in the network server or communication will cause the replicator to raise an error and replication to go `OFFLINE` [195].
- The network server must be running before the replicator is started. If the network server cannot be found, replication will go `OFFLINE` [195].

Correct operation requires building a suitable network filter using the defined protocol, and creating the JSON configuration file. A sample filter is provided for reference.

## 11.4.28.1. Network Client Configuration

The format of the configuration file defines the translation operation to be requested from the network client, in addition to the schema, table and column name. The format for the file is JSON, with the top-level hash defining the operation, and an array of field selections for each field that should be processed accordingly. For example:

```
{
   "String_to_HEX_v1" : [
      {
         "table" : "hextable",
         "schema" : "hexdb",
         "columns" : [
            "hexcol"
         ]
      }
   ]
}
```

The operation in this case is `String_to_HEX_v1`; this will be sent to the network server as part of the request. The column definition follows.

To send multiple columns from different tables to the same translation:

```
{
   "String_to_HEX_v1" : [
      {
         "table" : "hextable",
         "schema" : "hexdb",
         "columns" : [
            "hexcol"
         ]
      },
      {
         "table" : "hexagon",
         "schema" : "sourcetext",
         "columns" : [
            "itemtext"
         ]
      }
   ]
}
```

Alternatively, to configure different operations for the same two tables:

```
{
   "String_to_HEX_v1" : [
      {
         "table" : "hextable",
         "schema" : "hexdb",
         "columns" : [
            "hexcol"
         ]
      }
   ],
   "HEX_to_String_v1" : [
      {
         "table" : "hexagon",
         "schema" : "sourcetext",
         "columns" : [
            "itemtext"
         ]
      }
   ]
```

```
}
```

## 11.4.28.2. Network Filter Protocol

The network filter protocol has been designed to be both lightweight and binary data compatible, as it is designed to work with data that may be heavily encoded, binary, or compressed in nature.

The protocol operates through a combined JSON and optional binary payload structure that communicates the information. The JSON defines the communication type and metadata, while the binary payload contains the raw or translated information.

The filter communicates with the network server using the following packet types:

- prepare

  The `prepare` message is called when the filter goes online, and is designed to initialize the connection to the network server and confirm the supported filter types and operation. The format of the connection message is:

  ```
  {
     "payload" : -1,
     "type" : "prepare",
     "service" : "firstrep",
     "protocol" : "v0_9"
  }
  ```

  Where:

  - `protocol`

    The protocol version.

  - `service`

    The name of the replicator service that called the filter.

  - `type`

    The message type.

  - `payload`

    The size of the payload; a value of -1 indicates that there is no payload.

  The format of the response should be a JSON object and payload with the list of supported filter types in the payload section. The payload immediately follows the JSON, with the size of the list defined within the `payload` field of the returned JSON object:

  ```
  {
     "payload" : 22,
     "type" : "acknowledged",
     "protocol" : "v0_9",
     "service" : "firstrep",
     "return" : 0
  }Perl_BLOB_to_String_v1
  ```

  Where:

  - `protocol`

    The protocol version.

  - `service`

    The name of the replicator service that called the filter.

  - `type`

    The message type; when acknowledging the original prepare request it should be `acknowledge`.

  - `return`

    The return value. A value of 0 (zero) indicates no faults. Any true value indicates there was an issue.

  - `payload`

    The length of the appended payload information in bytes. This is used by the filter to identify how much additional data to read after the JSON object has been read.

The payload should be a comma-separated list of the supported transformation types within the network server.

- filter

  The `filter` message type is sent by Tungsten Replicator for each value from the replication stream that needs to be filtered and translated in some way. The format of the request is a JSON object with a trailing block of data, the payload, that contains the information to be filtered. For example:

```
{
   "schema" : "hexdb",
   "transformation" : "String_to_HEX_v1",
   "service" : "firstrep",
   "type" : "filter",
   "payload" : 22,
   "row" : 0,
   "column" : "hexcol",
   "table" : "hextable",
   "seqno" : 145196,
   "fragments" : 1,
   "protocol" : "v0_9",
   "fragment" : 1
}48656c6c6f20576f726c64
```

  Where:

  - `protocol`

    The protocol version.

  - `service`

    The service name the requested the filter.

  - `type`

    The message type, in this case, `filter`.

  - `row`

    The row of the source information from the THL that is being filtered.

  - `schema`

    The schema of the source information from the THL that is being filtered.

  - `table`

    The table of the source information from the THL that is being filtered.

  - `column`

    The column of the source information from the THL that is being filtered.

  - `seqno`

    The sequence number of the event from the THL that is being filtered.

  - `fragments`

    The number of fragments in the THL that is being filtered.

  - `fragment`

    The fragment number within the THL that is being filtered. The fragments may be sent individually and sequentially to the network server, so they may need to be retrieved, merged, and reconstituted depending on the nature of the source data and the filter being applied.

  - `transformation`

    The transformation to be performed on the supplied payload data. A single network server can support multiple transformations, so this information is provided to perform the corrupt operation. The actual transformation to be performed is taken from the JSON configuration file for the filter.

  - `payload`

The length, in bytes, of the payload data that will immediately follow the JSON filter request..

The payload that immediately follows the JSON block is the data from the column that should be processed by the network filter.

The response package should contain a copy of the supplied information from the requested filter, with the `payload` size updated to the size of the returned information, the message type changed to `filtered`, and the payload containing the translated data. For example:

```
{
   "transformation" : "String_to_HEX_v1",
   "fragments" : 1,
   "type" : "filtered",
   "fragment" : 1,
   "return" : 0,
   "seqno" : 145198,
   "table" : "hextable",
   "service" : "firstrep",
   "protocol" : "v0_9",
   "schema" : "hexdb",
   "payload" : 8,
   "column" : "hexcol",
   "row" : 0
}FILTERED
```

## 11.4.28.3. Sample Network Client

The following sample network server script is written in Perl, and is designed to translated packed hex strings (two-hex characters per byte) from their hex representation into their character representation.

```
#!/usr/bin/perl

use Switch;
use IO::Socket::INET;
use JSON qw( decode_json encode_json);
use Data::Dumper;

# auto-flush on socket
$| = 1;

my $serverName = "Perl_BLOB_to_String_v1";

while(1)
{
# creating a listening socket
my $socket = new IO::Socket::INET (
    LocalHost => '0.0.0.0',
    LocalPort => '3112',
    Proto => 'tcp',
    Listen => 5,
    Reuse => 1
);
die "Cannot create socket $!\n" unless $socket;
print "********\nServer waiting for client connection on port 3112\n******\n\n\n";

# Waiting for a new client connection
my $client_socket = $socket->accept();

# Fet information about a newly connected client
my $client_address = $client_socket->peerhost();
my $client_port = $client_socket->peerport();
print "Connection from $client_address:$client_port\n";

my $data = "";

while(    $data = $client_socket->getline())
{
    # Eead up to 1024 characters from the connected client
    chomp($data);

    print "\n\nReceived: <$data>\n";

    # Decode the JSON part
    my $msg = decode_json($data);

    # Extract payload
    my $payload = undef;

    if ($msg->{payload} > 0)
        {
            print STDERR "Reading $msg->{payload} bytes\n";
            $client_socket->read($payload,$msg->{payload});
            print "Payload: <$payload>\n";
```

```
          }

    switch( $msg->{'type'} )
    {
        case "prepare"
        {
              print STDERR "Received prepare request\n";

            # Send acknowledged message
            my $out = '{ "protocol": "v0_9", "type": "acknowledged", ' .
                      '"return": 0, "service": "' . $msg->{'service'} . '", "payload": ' .
                      length($serverName) . '}' . "\n" . $serverName;

            print $client_socket "$out";
            print "Sent: <$out>\n";
                print STDERR "Sent acknowledge request\n";
        }
        case "release"
        {
            # Send acknowledged message
            my $out = '{ "protocol": "v0_9", "type": "acknowledged", ' .
                      '"return": 0, "service": "' . $msg->{'service'} . '", "payload": 0}';

            print $client_socket "$out\n";
            print "Sent: <$out>\n";
        }
        case "filter"
        {
            # Send filtered message

                  print STDERR "Sending filtered payload\n";

            my $filtered = "FILTERED";
            my $out = <<END;
{
"protocol": "v0_9",
"type": "filtered",
"transformation": "$msg->{'transformation'}",
"return": 0,
"service": "$msg->{'service'}",
"seqno": $msg->{'seqno'},
"row": $msg->{'row'},
"schema": "$msg->{'schema'}",
"table": "$msg->{'table'}",
"column": "$msg->{'column'}",
"fragment": 1,
"fragments": 1,
"payload": @{[length($filtered)]}
}
END

$out =~ s/\n//g;
            print "About to send: <$out>\n";
            $client_socket->send("$out\n" . $filtered);
print("Response sent\n");
        }
    }

print("End of loop, hoping for next packet\n");
}

    # Notify client that we're done writing
    shutdown($client_socket, 1);

$socket->close();
}
```

## 11.4.29. `nocreatedbifnotexists.js` Filter

The `nocreatedbifnotexists` filter removes statements that start with:

```
CREATE DATABASE IF NOT EXISTS
```

| | |
|---|---|
| Pre-configured filter name | `nocreatedbifnotexists` |
| JavaScript Filter File | `tungsten-replicator/support/filters-javascript/nocreatedbifnotexists.js` |
| Property prefix | *`replicator.filter.nocreatedbifnotexists`* |
| Stage compatibility | `q-to-dbms` |
| tpm Option compatibility | `--svc-applier-filters` [424] |

| | | | |
|---|---|---|---|
| Data compatibility | | Any event | |
| Parameters | | | |
| Parameter | Type | Default | Description |

This can be useful in heterogeneous replication where Tungsten Replicator specific databases need to be removed from the replication stream.

The filter works in two phases. The first phase creates a global variable within the `prepare()` [492] function that defines the string to be examined:

```
function prepare()
{
  beginning = "CREATE DATABASE IF NOT EXISTS";
}
```

Row based changes can be ignored, but for statement based events, the SQL is examine and the statement removed if the SQL starts with the text in the `beginning` variable:

```
sql = d.getQuery();
if(sql.startsWith(beginning))
{
    data.remove(i);
    i--;
}
```

## 11.4.30. OptimizeUpdates Filter

The `optimizeupdates` filter works with row-based events to simplify the update statement and remove columns/values that have not changed. This reduces the workload and row data exchanged between replicators.

| | |
|---|---|
| Pre-configured filter name | `optimizeupdates` |
| Classname | `com.continuent.tungsten.replicator.filter.OptimizeUpdatesFilter` |
| Property prefix | *`replicator.filter.optimizeupdates`* |
| Stage compatibility | |
| tpm Option compatibility | |
| Data compatibility | Row events |
| Parameters | |
| (none) | |

The filter operates by removing column values for keys in the update statement that do not change. For example, when replicating the row event from the statement:

```
mysql>  update testopt set msg = 'String1', string = 'String3' where id = 1;
```

Generates the following THL event data:

```
- SQL(0) =
 - ACTION = UPDATE
 - SCHEMA = test
 - TABLE = testopt
 - ROW# = 0
  - COL(1: id) = 1
  - COL(2: msg) = String1
  - COL(3: string) = String3
  - KEY(1: id) = 1
```

Column 1 (`id`) in this case is automatically implied by the KEY entry required for the update.

With the `optimizeupdates` filter enabled, the data in the THL is simplified to:

```
- SQL(0) =
 - ACTION = UPDATE
 - SCHEMA = test
 - TABLE = testopt
 - ROW# = 0
  - COL(2: msg) = String1
  - COL(3: string) = String4
  - KEY(1: id) = 1
```

In tables where there are multiple keys the stored THL information can be reduced further.

### Warning

The filter works by comparing the value of each KEY and COL entry in the THL and determining whether the value has changed or not. If the number of keys and columns do not match then the filter will fail with the following error message:

```
Caused by: java.lang.Exception: Column and key count is different in this event! Cannot filter
```

This may be due to a filter earlier within the filter configuration that has optimized or simplified the data. For example, the `pkey` filter removes KEY entries from the THL that are not primary keys, or `dropcolumn` which drops column data.

The following error message may appear in the logs and in the output from trepctl status to indicate that this ordering issue may be the problem:

```
OptimizeUpdatesFilter cannot filter, because column and key count is different.
Make sure that it is defined before filters which remove keys (eg. PrimaryKeyFilter).
```

## 11.4.31. PrimaryKey Filter

The `PrimaryKey` filter adds primary key information to row-based replication data. This is required by heterogeneous environments to ensure that the primary key is identified when updating or deleting tables. Without this information, the primary key to use, for example as the document ID in a document store such as MongoDB, is generated dynamically. In addition, without this filter in place, when performing update or delete operations a full table scan is performed on the target dataserver to determine the record that must be updated.

| | |
|---|---|
| Pre-configured filter name | `pkey` |
| Classname | `com.continuent.tungsten.replicator.filter.PrimaryKeyFilter` |
| Property prefix | *`replicator.filter.pkey`* |
| Stage compatibility | `binlog-to-q` |
| tpm Option compatibility | `--repl-svc-extractor-filters` [425] |
| Data compatibility | Row events |
| Keeps Cached Data | Yes |
| Cached Refreshed When? | Emptied when going `OFFLINE` [195]; Updated when `ALTER` statement seen |
| Metadata Updated | Yes; `tungsten_filter_primarykey=true` |

Parameters

| Parameter | Type | Default | Description |
|---|---|---|---|
| *`user`* | string | *`${replicator.global.extract.db.user}`* | The username for the connection to the database for looking up column definitions |
| *`password`* | string | *`${replicator.global.extract.db.password}`* | The password for the connection to the database for looking up column definitions |
| *`url`* | string | *`jdbc:mysql:thin://${replicator.global.extract.db.host}:» ${replicator.global.extract.db.port}/${replicator.schema}?createDB=true`* | JDBC URL of the database connection to use for looking up column definitions |
| *`addPkeyToInsert`* | `boolean` | `false` | If set to true, primary keys are added to `INSERT` operations. This setting is required for batch loading |
| *`addColumnsToDeletes`* | `boolean` | `false` | If set to true, full column metadata is added to `DELETE` operations. This setting is required for batch loading |
| *`custompkey`* | `filename` | `(empty)` | If set to the value of a JSON file, the file is used to determine custom primary key specifications in place of database derived versions. See Section 11.4.31.1, "Setting Custom Primary Key Definitions". |
| *`reconnectTimeout`* | integer | 3600 | Timeout for refreshing connection to database |

### Note

This filter is designed to be used for testing and with heterogeneous replication where the field name information can be used to construct and build target data structures.

For example, in the following THL fragment, the key information includes data for all columns, which the is the default behavior for `UPDATE` and `DELETE` operations.

```
SEQ# = 142 / FRAG# = 0 (last frag)
- TIME = 2013-08-01 19:31:04.0
- EPOCH# = 122
- EVENTID = mysql-bin.000012:0000000000022187;0
- SOURCEID = host31
- METADATA = [mysql_server_id=1;dbms_type=mysql;service=alpha;shard=test]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1]
- SQL(0) =
 - ACTION = UPDATE
 - SCHEMA = test
 - TABLE = salesadv
 - ROW# = 0
  - COL(1: id) = 2
  - COL(2: country) = 1
  - COL(3: city) = 8374
  - COL(4: salesman) = 1
  - COL(5: value) = 89000.00
  - KEY(1: id) = 2
  - KEY(2: country) = 1
  - KEY(3: city) = 8374
  - KEY(4: salesman) = 1
  - KEY(5: value) = 89000.00
```

When the `PrimaryKey` filter is enabled, the key information has been optimized to only contain the actual primary keys in the row-based THL record:

```
SEQ# = 142 / FRAG# = 0 (last frag)
- TIME = 2013-08-01 19:31:04.0
- EPOCH# = 122
- EVENTID = mysql-bin.000012:0000000000022187;0
- SOURCEID = host31
- METADATA = [mysql_server_id=1;dbms_type=mysql;service=alpha;shard=test]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1]
- SQL(0) =
 - ACTION = UPDATE
 - SCHEMA = test
 - TABLE = salesadv
 - ROW# = 0
  - COL(1: id) = 2
  - COL(2: country) = 1
  - COL(3: city) = 8374
  - COL(4: salesman) = 1
  - COL(5: value) = 89000.00
  - KEY(1: id) = 2
```

The final line shows the addition of the primary key `id` added to THL event.

### Important

The filter determines primary key information by examining the DDL for the table, and keeping that information in an internal cache. If the DDL for a table is not known, or an `ALTER TABLE` statement is identified, the cache information is updated before the THL is then modified with the primary key information.

In the situation where you enable the filter, but have not create primary key information on the tables, it is possible that creating or adding other index types (such as `UNIQUE`) on a table, could lead to the incorrect primary key information being updated in the THL, particularly if there are active transactions taking place during and/or immediately after the `ALTER` statement.

The safest way to perform an index update in case remains the same as for any safe DDL update:

- Put the replicator offline
- Change the DDL for the table or tables
- Put the replicator online

The two options, *`addPkeyToInsert`* and *`addColumnsToDeletes`* add the primary key information to `INSERT` and `DELETE` operations respectively. In a heterogeneous environment, these options should be enabled to prevent full-table scans during update and deletes.

## 11.4.31.1. Setting Custom Primary Key Definitions

6.0.0 or later. Custom primary key configuration support available in 6.0.0 or later.

Not all tables and databases set or provide explicit primary key information, and in some cases, it is not possible to change the index definition on the source table to include primary key information. Without primary key information in the THL, replicating data into heterogeneous

targets can fail, because there is no way for the target environment to correctly identify the primary key information, and therefore the specific record or records to be updated.

The `pkey` filter supports defining custom columns to make up a primary key in this situation, avoiding the need to explicitly set index information within the database.

The custom primary key setting works as follows:

- When processing a THL entry, if the custom primary key configuration file has been set and exists, and the incoming schema/table name has been defined in the configuraiton file, the custom pkey configuration is used.
- Otherwise, the primary key information is taken from the database if it exists.

If neither of these conditions is met, then no primary key data is added to the THL during process.

To configure a custom primary key for one or more tables:

1. Copy the sample primary key configuration file, located within the distribution as `tungsten/tungsten-replicator/support/filters-config/custompkey.json` into the `share` directory within the installation. For example, `/opt/continuent/share`.
2. Update the configuration to include the specific primary key settings for the incoming table. The format of the file is JSON, using a structured layout based on the common JSON filter configuration format (see Section 11.5, "Standard JSON Filter Configuration"). The sample file contains an example for the schema `test` and the table `msg`:

```
{
    "__default": {
        "IGNORE" : "pkey"
    },
    "test" : {
        "msg" : {
            "msg" : "pkey"
        }
    }
}
```

   In the above example, `msg` is the name of the column to be specified as the primary key. The `pkey` indicates that this column must be a primary key field. You can also specify multiple columns:

```
{
    "__default": {
        "IGNORE" : "pkey"
    },
    "test" : {
        "msg" : {
            "id"  : "pkey",
            "msg" : "pkey"
        }
    }
}
```

   To include another table within the same schema:

```
{
    "__default": {
        "IGNORE" : "pkey"
    },
    "test" : {
        "msg" : {
            "msg" : "pkey"
        },
        "orders" : {
            "orderid" : "pkey"
        }
    }
}
```

   **Note**

   The `__default` section must remain in place, although it has no impact on processing, it is used to ensure the file is valid for the filter configuration.

3. Update the configuration of your installtion through tpm to specify the custom primary key file in the properties:

```
shell> tpm update alpha --property=replicator.filter.pkey.custompkey=/opt/continuent/share/custompkey.json
```

Once this process has been completed, the replicator will add the custom primary key fields to the THL during processing. For example:

```
- SQL(0) =
```

```
- ACTION = INSERT
- SCHEMA = test
- TABLE = msg
- ROW# = 0
 - COL(1: id) = 6
 - COL(2: msg) = new test
 - KEY(1: id) = NULL
```

## 11.4.32. PrintEvent Filter

Outputs transaction event information to the replication logging system

| | |
|---|---|
| Pre-configured filter name | `printevent` |
| Classname | `com.continuent.tungsten.replicator.filter.PrintEventFilter` |
| Property prefix | *`replicator.filter.printevent`* |
| Stage compatibility | |
| tpm Option compatibility | |
| Data compatibility | Any event |
| Parameters | |
| (none) | |

## 11.4.33. Rename Filter

The `rename` filter enables schemas to be renamed at the database, table and column levels, and for complex combinations of these renaming operations. Configuration is through a CSV file that defines the rename parameters. A single CSV file can contain multiple rename definitions. The rename operations occur only on ROW based events.

The `rename` filter also performs schema renames on statement data.

| | | | |
|---|---|---|---|
| Pre-configured filter name | | `rename` | |
| Classname | | `com.continuent.tungsten.replicator.filter.RenameFilter` | |
| Property prefix | | *`replicator.filter.rename`* | |
| Stage compatibility | | | |
| tpm Option compatibility | | | |
| Data compatibility | | Row events. | |
| Parameters | | | |
| Parameter | Type | Default | Description |
| *`definitionsFile`* | string | `{replicator.home.dir}/samples/extensions/java/rename.csv` | Location of the CSV file that contains the rename definitions. |

The CSV file is only read when an explicit reconfigure operation is triggered. If the file is changed, a configure operation (using tpm update) must be initiated to force reconfiguration.

To enable using the default CSV file:

```
shell> ./tools/tpm update alpha --svc-applier-filters=rename
```

The CSV consists of multiple lines, one line for each rename specification. Comments are supposed using the `#` character.

The format of each line of the CSV is:

```
originalSchema,originalTable,originalColumn,newSchema,newTable,newColumn
```

Where:

- `originalSchema`, `originalTable`, `originalColumn` define the original schema, table and column.

  Definition can either be:

  - Explicit schema, table or column name
  - `*` character, which indicates that all entries should match.

- `newSchema`, `newTable`, `newColumn` define the new schema, table and column for the corresponding original specification.

  Definition can either be:

  - Explicit schema, table or column name
  - `-` character, which indicates that the corresponding object should not be updated.

For example, the specification:

```
*,chicago,*,-,newyork,-
```

Would rename the table `chicago` in every database schema to `newyork`. The schema and column names are not modified.

The specification:

```
*,chicago,destination,-,-,source
```

Would match all schemas, but update the column `destination` in the table `chicago` to the column name `source`, without changing the schema or table name.

Processing of the individual rules is executed in a specific order to allow for complex matching and application of the rename changes.

- Rules are case sensitive.
- Schema names are looked up in the following order:
  1. `schema.table` (explicit schema/table)
  2. `schema.*` (explicit schema, wildcard table)
- Table names are looked up in the following order:
  1. `schema.table` (explicit schema/table)
  2. `*.table` (wildcard schema, explicit table)
- Column names are looked up in the following order:
  1. `schema.table` (explicit schema/table)
  2. `schema.*` (explicit schema, wildcard table)
  3. `*.table` (wildcard schema, explicit table)
  4. `*.*` (wildcard schema, wildcard table)
- Rename operations match the first specification according to the above rules, and only one matching rule is executed.

### 11.4.33.1. Rename Filter Examples

When processing multiple entries that would match the same definition, the above ordering rules are applied. For example, the definition:

```
asia,*,*,america,-,-
asia,shanghai,*,europe,-,-
```

Would rename `asia.shanghai` to `europe.shanghai`, while renaming all other tables in the schema `asia` to the schema `america`. This is because the explicit `schema.table` rule is matched first and then executed.

Complex renames involving multiple schemas, tables and columns can be achieved by writing multiple rules into the same CSV file. For example given a schema where all the tables currently reside in a single schema, but must be renamed to specific continents, or to a 'miscellaneous' schema, while also updating the column names to be more neutral would require a detailed rename definition.

Existing tables are in the schema `sales`:

```
chicago
newyork
london
paris
munich
moscow
tokyo
shanghai
sydney
```

Need to be renamed to:

```
northamerica.chicago
northamerica.newyork
europe.london
europe.paris
europe.munich
misc.moscow
asiapac.tokyo
asiapac.shanghai
misc.sydney
```

Meanwhile, the table definition needs to be updated to support more complex structure:

```
id
area
country
city
value
type
```

The area is being updated to contain the region within the country, while the value should be renamed to the three-letter currency code, for example, the `london` table would rename the `value` column to `gbp`.

The definition can be divided up into simple definitions at each object level, relying on the processing order to handle the individual exceptions. Starting with the table renames for the continents:

```
sales,chicago,*,northamerica,-,-
sales,newyork,*,northamerica,-,-
sales,london,*,europe,-,-
sales,paris,*,europe,-,-
sales,munich,*,europe,-,-
sales,tokyo,*,asiapac,-,-
sales,shanghai,*,asiapac,-,-
```

A single rule to handle the renaming of any table not explicitly mentioned in the list above into the `misc` schema:

```
*,*,*,misc,-,-
```

Now a rule to change the `area` column for all tables to `region`. This requires a wildcard match against the schema and table names:

```
*,*,area,-,-,region
```

And finally the explicit changes for the value column to the corresponding currency:

```
*,chicago,value,-,-,usd
*,newyork,value,-,-,usd
*,london,value,-,-,gbp
*,paris,value,-,-,eur
*,munich,value,-,-,eur
*,moscow,value,-,-,rub
*,tokyo,value,-,-,jpy
*,shanghai,value,-,-,cny
*,sydney,value,-,-,aud
```

## 11.4.34. Replicate Filter

The `replicate` filter enables explicit inclusion or exclusion of tables and schemas. Each specification supports wildcards and multiple entries.

| | | | |
|---|---|---|---|
| Pre-configured filter name | | `replicate` | |
| Classname | | `com.continuent.tungsten.replicator.filter.ReplicateFilter` | |
| Property prefix | | *`replicator.filter.replicate`* | |
| Stage compatibility | | Any | |
| tpm Option compatibility | | | |
| Data compatibility | | Any event | |
| Parameters | | | |
| Parameter | Type | Default | Description |
| *`ignore`* | string | empty | Comma separated list of database/tables to ignore during replication |
| *`do`* | string | empty | Comma separated list of database/tables to replicate |

Rules using the supplied parameters are evaluated as follows:

- When both `do` and `ignore` are empty, updates are allowed to any table.
- When only `do` is specified, only the schemas (or schemas and tables) mentioned in the list are replicated.
- When only `ignore` is specified, all schemas/tables are replicated except those defined.

For each parameter, a comma-separated list of schema or schema and table definitions are supported, and wildcards using `*` (any number of characters) and `?` (single character) are also honored. For example:

- `do=sales`

  Replicates only the schema `sales`.

- `ignore=sales`

  Replicates everything, ignoring the schema `sales`.

- `ignore=sales.*`

  Replicates everything, ignoring the schema `sales`.

- `ignore=sales.quarter?`

  Replicates everything, ignoring all tables within the `sales` schema starting with `sales.quarter` and a single character. This would ignore `sales.quarter1` but replicate `sales.quarterlytotals`.

- `ignore=sales.quarter*`

  Replicates everything, ignoring all tables in the schema `sales` starting with `quarter`.

- `do=*.quarter`

  Replicates only the table named `quarter` within any schema.

- `do=sales.*totals,invoices`

  Replicates only tables in the `sales` schema that end with `totals`, and the entire `invoices` schema.

## 11.4.35. ReplicateColumns Filter

Removes selected columns from row-based transaction data.

| | | | |
|---|---|---|---|
| Pre-configured filter name | | `replicatecolumns` | |
| Classname | | `com.continuent.tungsten.replicator.filter.ReplicateColumnsFilter` | |
| Property prefix | | *`replicator.filter.replicatecolumns`* | |
| Stage compatibility | | | |
| tpm Option compatibility | | | |
| Data compatibility | | Row events | |
| Parameters | | | |
| Parameter | Type | Default | Description |
| *`ignore`* | string | empty | Comma separated list of tables and optional columns names to ignore during replication |
| *`do`* | string | empty | Comma separated list of tables and optional column names to replicate |

## 11.4.36. Row Add Database Name Filter

The `rowadddbname` filter adds a new column to every incoming row of data containing the schema name of the table. This can be used in combination with analytics replication targets where the information is being written into a single schema, concentrating data from multiple source databases into a single database. Optionally, the filter can also add the information as a primary key (required for some heterogeneous targets), and the hash of the source database name.

| | |
|---|---|
| Pre-configured filter name | `rowadddbname` |
| Classname | `com.continuent.tungsten.replicator.filter.JavaScriptFilter` |

| | | | |
|---|---|---|---|
| Property prefix | | *replicator.filter.rowadddbname* | |
| Stage compatibility | | Any | |
| tpm Option compatibility | | | |
| Data compatibility | | Row-events only | |
| Parameters | | | |
| Parameter | Type | Default | Description |
| *adddbhash* | boolean | true | Add a hash column, computed using the Java hash value of the string for the incoming source database name |
| *addkey* | boolean | true | Add the information to the primary key data in the THL, in addition to the column data |
| *fieldname* | string | dbname | The name of the database name column that will be added |
| *fieldhashname* | string | dbname | The name of the database has name column that will be added |

The `rowadddbname` filter adds a column to every row of THL processed by the filter. This can be used when data is being written into a single target schema within an analytics environment, and where the source database can be used to identify a customer, project or dataset, and therefore queried within the analytics platform either specifically or with other datasets.

The filter is able to perform the following modifications to every row of incoming data:

- Add the source database or schema name.
- Add a numerical hash value of the string of the source database of schema name.
- Add the database name (and hash name) to the primary key data.

For example, the source THL:

```
- ROW# = 0
  - COL(1: id) = 12
  - COL(2: msg) = Hello
  - COL(3: msg2) = World
  - KEY(1: id) = NULL
```

And after the filter has been applied:

```
- ROW# = 0
  - COL(1: id) = 12
  - COL(2: msg) = Hello
  - COL(3: msg2) = World
  - COL(4: dbname) = msg
  - COL(5: dbname_hash) = 108417
  - KEY(1: id) = NULL
  - KEY(4: dbname) = NULL
  - KEY(5: dbname_hash) = NULL
```

This filter is a required component of deployments when replicating into a single schema within Vertica (see Section 4.3, "Deploying the Vertica Applier") and Amazon Redshift (see Section 4.2, "Deploying the Amazon Redshift Applier").

## 11.4.37. Row Add Transaction Info Filter

The `rowaddtxninfo` filter examines an entire row-based event within the THL and then builds a list of transaction information which is embedded into the metadata for the event. This can be combined with functionality in corresponding appliers, such as Section 4.4, "Deploying the Kafka Applier" which can then be used and embedded into messages or document-based databases.

| | |
|---|---|
| Pre-configured filter name | `rowaddtxninfo` |
| Classname | `com.continuent.tungsten.replicator.filter.JavaScriptFilter` |
| Property prefix | *replicator.filter.rowaddtxninfo* |
| Stage compatibility | Any |
| tpm Option compatibility | |
| Data compatibility | Row-events only |
| Parameters | |

| Parameter | Type | Default | Description |
|---|---|---|---|

The `rowaddtxninfo` filter processes an entire transaction to determine the following information:

- List of schemas and tables affected by the transaction.
- A count of the rows inserted, updated, or deleted per schema/table combination.
- A total count of all the rows inserted, updated, or deleted across the entire transaction.

Once the information about the transaction has been collected, the data is then formulated into key/value pairs that are incorporated into the metadata for the entire THL event.

For example, when inserting information into the same table across three separate statements the but the same overall transaction, the THL with the filter enabled looks like the example below:

```
shell> thl list -last
SEQ# = 162 / FRAG# = 0 (last frag)
- TIME = 2018-03-02 08:47:14.0
- EPOCH# = 162
- EVENTID = mysql-bin.000065:0000000000000534;-1
- SOURCEID = trfiltera
- METADATA = [mysql_server_id=366;dbms_type=mysql;tz_aware=true;strings=utf8;service=alpha;shard=msg;
» tungsten_filter_columnname=true; tungsten_filter_primarykey=true;tungsten_filter_enumtostring=true;
» txinfo_rowcount_msg.msg=1;txinfo_rowcount_msg.msgsub=2;txinfo_rowcount=3]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1, time_zone = '+00:00']
- SQL(0) =
 - ACTION = INSERT
 - SCHEMA = msg
 - TABLE = msg
 - ROW# = 0
  - COL(1: id) = 108
  - COL(2: msg) = txinfo
  - COL(3: msg2) = txinfo
  - KEY(1: id) = NULL
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1, time_zone = '+00:00']
- SQL(1) =
 - ACTION = INSERT
 - SCHEMA = msg
 - TABLE = msgsub
 - ROW# = 0
  - COL(1: id) = 108
  - COL(2: msg) = subtx
 - ROW# = 1
  - COL(1: id) = 108
  - COL(2: msg) = subxt
```

Examining the metadata more closely you can see the transaction information:

```
...   txinfo_rowcount_msg.msg=1;txinfo_rowcount_msg.msgsub=2;txinfo_rowcount=3 ...
```

This can be further extrapolated as:

- `txinfo_rowcount_msg.msg=1`

  One row has been updated in the `msg.msg` schema/table.

- `txinfo_rowcount_msg.msgsub=2`

  Two rows has been updated in the `msg.submsg` schema/table.

- `txinfo_rowcount=3`

  A total of three rows have been updated within the transaction.

> **Note**
>
> If you transactional information is needed within a Kafka deployment, this filter must have been enabled for the transaction information to be included. See Section 4.4.2.1, “Optional Configuration Parameters for Kafka”, for more information.

## 11.4.38. SetToString Filter

The `SetToString` converts the `SET` column type from the internal representation to a string-based representation in the THL. This achieved by accessing the extractor database, obtaining the table definitions, and modifying the THL data before it is written into the THL file.

| | | | |
|---|---|---|---|
| Pre-configured filter name | | settostring | |
| Classname | | com.continuent.tungsten.replicator.filter.SetToStringFilter | |
| Property prefix | | *replicator.filter.settostring* | |
| Stage compatibility | | binlog-to-q | |
| tpm Option compatibility | | --repl-svc-extractor-filters [425] | |
| Data compatibility | | Row events | |
| Parameters | | | |
| Parameter | Type | Default | Description |
| *user* | string | *${replicator.global.extract.db.user}* | The username for the connection to the database for looking up column definitions |
| *password* | string | *${replicator.global.extract.db.password}* | The password for the connection to the database for looking up column definitions |
| *url* | string | jdbc:mysql:thin://${replicator.global.extract.db.host}: » ${replicator.global.extract.db.port}/${replicator.schema}?createDB=true | JDBC URL of the database connection to use for looking up column definitions |

The SetToString filter should be used with heterogeneous replication to ensure that the data is represented as the string value, not the internal numerical representation.

In the THL output below, the table has a SET column, salesman:

```
mysql> describe salesadv;
+----------+-----------------------------------+------+-----+---------+----------------+
| Field    | Type                              | Null | Key | Default | Extra          |
+----------+-----------------------------------+------+-----+---------+----------------+
| id       | int(11)                           | NO   | PRI | NULL    | auto_increment |
| country  | enum('US','UK','France','Australia') | YES  |     | NULL    |                |
| city     | int(11)                           | YES  |     | NULL    |                |
| salesman | set('Alan','Zachary')             | YES  |     | NULL    |                |
| value    | decimal(10,2)                     | YES  |     | NULL    |                |
+----------+-----------------------------------+------+-----+---------+----------------+
```

When extracted in the THL, the representation uses the internal value (for example, 1 for the first element of the set description). This can be seen in the THL output below.

```
SEQ# = 138 / FRAG# = 0 (last frag)
- TIME = 2013-08-01 19:09:35.0
- EPOCH# = 122
- EVENTID = mysql-bin.000012:0000000000021434;0
- SOURCEID = host31
- METADATA = [mysql_server_id=1;dbms_type=mysql;service=alpha;shard=test]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1]
- SQL(0) =
 - ACTION = INSERT
 - SCHEMA = test
 - TABLE = salesadv
 - ROW# = 0
  - COL(1: id) = 2
  - COL(2: country) = 1
  - COL(3: city) = 8374
  - COL(4: salesman) = 1
  - COL(5: value) = 35000.00
```

For the salesman column, the corresponding value in the THL is 1. With the SetToString filter enabled, the value is expanded to the corresponding string value:

```
SEQ# = 121 / FRAG# = 0 (last frag)
- TIME = 2013-08-01 19:05:14.0
- EPOCH# = 102
- EVENTID = mysql-bin.000012:0000000000018866;0
- SOURCEID = host31
- METADATA = [mysql_server_id=1;dbms_type=mysql;service=alpha;shard=test]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1]
- SQL(0) =
 - ACTION = INSERT
 - SCHEMA = test
 - TABLE = salesadv
 - ROW# = 0
  - COL(1: id) = 1
```

```
  - COL(2: country) = US
  - COL(3: city) = 8374
  - COL(4: salesman) = Alan
  - COL(5: value) = 35000.00
```

The examples here also show the Section 11.4.20, "EnumToString Filter" and Section 11.4.5, "ColumnName Filter" filters.

## 11.4.39. Shard Filter

Used to enforce database schema sharding between specific Primaries

| | | | |
|---|---|---|---|
| Pre-configured filter name | | `shardfilter` | |
| Classname | | `com.continuent.tungsten.replicator.filter.ShardFilter` | |
| Property prefix | | *`replicator.filter.shardfilter`* | |
| Stage compatibility | | | |
| tpm Option compatibility | | | |
| Data compatibility | | Any event | |
| Parameters | | | |
| Parameter | Type | Default | Description |
| *`enabled`* | `boolean` | `false` | If set to true, enables the shard filter |
| *`unknownShardPolicy`* | string | error | Select the filter policy when the shard unknown; valid values are `accept`, `drop`, `warn`, and `error` |
| *`unwantedShardPolicy`* | string | error | Select the filter policy when the shard is unwanted; valid values are `accept`, `drop`, `warn`, and `error` |
| *`enforcedHome`* | `boolean` | `false` | If true, enforce the home for the shard |
| *`allowWhitelisted`* | `boolean` | `false` | If true, allow explicitly whitelisted shards |
| *`autoCreate`* | `boolean` | `false` | If true, allow shard rules to be created automatically |

## 11.4.40. `shardbyrules.js` Filter

The `shardbyrules` filter allows you to specify granular schema and table level rules for sharding of transactions throguh the replicator. This can provide enhanced performance where regular schema only based sharding would not suit the profile of your application.

| | | | |
|---|---|---|---|
| Pre-configured filter name | | `shardbyrules` | |
| JavaScript Filter File | | `tungsten-replicator/support/filters-javascript/shardbyrules.js` | |
| Property prefix | | *`replicator.filter.shardbyrules`* | |
| Stage compatibility | | `q-to-dbms` | |
| tpm Option compatibility | | `--svc-applier-filters` [424] | |
| Data compatibility | | ROW | |
| Parameters | | | |
| Parameter | Type | Default | Description |
| *`definitionsFile`* | string | `support/filters-config/shards.json` | JSON file containing the definition of which events and which tables to skip |

### Note

For this filter to function, you will need to ensure your database is configured for ROW based binary logging

The key part of the filter is configuring the rules to suit your sharding requirements. Start by copying the sample file and then editing to suit:

```
shell> cp /opt/continuent/tungsten/tungsten-replicator/support/filters-config/shards.json /opt/continuent/share/shards.json
```

Within the configuration, you would then specify this definitionsFile:

```
svc-applier-filters=shardbyrules
property=replicator.filter.shardbyrules.definitionsFile=/opt/continuent/share/shards.json
```

Example:

```
{
  "default": "defaultShardName",
  "schemas": [
    {
      "schema": "schemaA",
      "shardId": "MyShard1"
    },
    {
      "schema": "schemaB",
      "shardId": "MyShard2"
    }
  ],
  "tables": [
    {
      "schema": "schemaB",
      "table": "MyTable1",
      "shardId": "MyShard1"
    }
  ]
}
```

- schemaA gets assigned to MyShard1
- schemaB gets assigned to MyShard2
- With the exception of schemaB.MyTable1, which also gets assigned to MyShard1.

With this behavior, it would be able to exceute concurrently with transactions either hitting other tables from schemaB, or with tables from other schemas.

## 11.4.41. `shardbyseqno.js` Filter

Shards within the replicator enable data to be parallelized when they are applied on the Target.

| | | | |
|---|---|---|---|
| Pre-configured filter name | | `shardbyseqno` | |
| JavaScript Filter File | | `tungsten-replicator/support/filters-javascript/shardbyseqno.js` | |
| Property prefix | | *`replicator.filter.shardbyseqno`* | |
| Stage compatibility | | `q-to-dbms` | |
| tpm Option compatibility | | `--svc-applier-filters` [424] | |
| Data compatibility | | Any event | |
| Parameters | | | |
| Parameter | Type | Default | Description |
| *`shards`* | `numeric` | (none) | Number of shards to be used by the applier |

The `shardbyseqno` filter updates the shard ID, which is embedded into the event metadata, by a configurable number of shards, set by the *`shards`* parameter in the configuration:

```
replicator.filter.shardbyseqno=com.continuent.tungsten.replicator.filter.JavaScriptFilter
replicator.filter.shardbyseqno.script=${replicator.home}/samples/extensions/javascript/shardbyseqno.js
replicator.filter.shardbyseqno.shards=10
```

The filter works by setting the shard ID in the event using the `setShardId()` method on the event object:

```
event.setShardId(event.getSeqno() % shards);
```

### Note

Care should be taken with this filter, as it assumes that the events can be applied in a completely random order by blindly updating the shard ID to a computed valued. Sharding in this way is best used when provisioning new Targets.

## 11.4.42. `shardbytable.js` Filter

An alternative to sharding by sequence number is to create a shard ID based on the individual database and table. The `shardbytable` filter achieves this at a row level by combining the schema and table information to form the shard ID. For all other events, including statement based events, the shard ID `#UNKNOWN` is used.

| | |
|---|---|
| Pre-configured filter name | `shardbytable` |

| | | | |
|---|---|---|---|
| JavaScript Filter File | | tungsten-replicator/support/filters-javascript/shardbytable.js | |
| Property prefix | | *replicator.filter.shardbytable* | |
| Stage compatibility | | remote-to-thl | |
| tpm Option compatibility | | --svc-remote-filters [425] | |
| Data compatibility | | ROW | |
| Parameters | | | |
| Parameter | Type | Default | Description |

### Note

For this filter to function, you will need to ensure your database is configured for ROW based binary logging

The key part of the filter is the extraction and construction of the ID, which occurs during row processing:

```
oneRowChange = rowChanges.get(j);
schemaName = oneRowChange.getSchemaName();
tableName  = oneRowChange.getTableName();

id = schemaName + "_" + tableName;
if (proposedShardId == null)
{
  proposedShardId = id;
}
```

## 11.4.43. SkipEventByType Filter

The SkipEventByType filter enables you to skip individual events based on the event type, schema and table. For example, if you want to skip all DELETE events on the schema/table SALES.INVOICES (to prevent deletion of invoice data), this filter will skip the event entirely and it will not be applied to the target.

| | | | |
|---|---|---|---|
| Pre-configured filter name | | skipeventbytype | |
| Classname | | com.continuent.tungsten.replicator.filter.SkipEventByTypeFilter | |
| Property prefix | | *replicator.filter.skipeventbytype* | |
| Stage compatibility | | any | |
| tpm Option compatibility | | --repl-svc-extractor-filters [425], --repl-svc-applier-filters [424] | |
| Data compatibility | | Row events | |
| Parameters | | | |
| Parameter | Type | Default | Description |
| *definitionsFile* | string | support/filters-config/skipeventbytype.json | JSON file containing the definition of which events and which tables to skip |

Configuration of the filter is made using the generic JSON file, which supports both default options to happen for all tables not otherwise explicitly specified. The default JSON file allows all operations:

```
{
    "__default": {
        "INSERT" : "allow",
        "DELETE" : "allow",
        "UPDATE" : "allow"
    },
    "SCHEMA" : {
        "TABLE" : {
            "INSERT" : "allow",
            "DELETE" : "deny",
            "UPDATE" : "deny"
        }
    }
}
```

The default section handles the default response when an explicit schema or table name does not appear. Further sections are then organised by schema and then table name. Where the setting is allow, the operation will be processed. A deny skips the entire event.

To disable all DELETE operations, regardless of which table they occur in:

```
{
```

```
    "__default": {
        "INSERT" : "allow",
        "DELETE" : "deny",
        "UPDATE" : "allow"
    },
    "SCHEMA" : {
        "TABLE" : {
            "INSERT" : "allow",
            "DELETE" : "deny",
            "UPDATE" : "deny"
        }
    }
}
```

To normally allow all operations, except on the SALES.INVOICE schema/table:

```
{
    "__default": {
        "INSERT" : "allow",
        "DELETE" : "allow",
        "UPDATE" : "allow"
    },
    "SALES" : {
        "INVOICE" : {
            "INSERT" : "allow",
            "DELETE" : "deny",
            "UPDATE" : "deny"
        }
    }
}
```

## 11.4.44. TimeDelay (delay) Filter

The `TimeDelayFilter` delays writing events to the THL and should be used only on Appliers in the `remote-to-thl` stage. This delays writing the transactions into the THL files, but allows the application of the data to the database to continue without further intervention.

From version 6.1.4 the delayInMs Filter was also added which allows delay prescion in milliseconds. See Section 11.4.45, "TimeDelayMsFilter (delayInMS) Filter" for more detail.

| | | | |
|---|---|---|---|
| Pre-configured filter name | | `delay` | |
| Classname | | `com.continuent.tungsten.replicator.filter.TimeDelayFilter` | |
| Property prefix | | *`replicator.filter.delay`* | |
| Stage compatibility | | remote-to-thl | |
| tpm Option compatibility | | `--repl-svc-thl-filters` [426] | |
| Data compatibility | | Any event | |
| Parameters | | | |
| Parameter | Type | Default | Description |
| *`delay`* | numeric | 300 | Number of seconds to delay transaction processing row |

The `TimeDelayFilter` delays the application of transactions recorded in the THL. The delay can be used to allow point-in-time recovery of DML operations before the transaction has been applied to the Target, or where data may need to be audited or checked before transactions are committed.

### Note

For effective operation, Source and Targets should be synchronized using NTP or a similar protocol.

To enable the `TimeDelayFilter`, update the tungsten.ini configuration file to enable the filter. For example, to enable the delay for 600 seconds:

```
shell> vi /etc/tungsten/tungsten.ini
[serviceName]
...
svc-applier-filters=delay
property=replicator.filter.delay.delay=600
...

shel> tpm update
```

Time delay of transaction events should be performed with care, since the delay will prevent a Target from being up to date compared to the Source. In the event of a node failure, an up to date Target is required to ensure that data is safe.

## 11.4.45. TimeDelayMsFilter (delayInMS) Filter

The `TimeDelayMsFilter` delays writing events to the THL and should be used only on Appliers in the `remote-to-thl` stage. This delays writing the transactions into the THL files, but allows the application of the data to the database to continue without further intervention.

This filter allows delay precision in milliseconds. If you wish to delay to second precision, then the `TimeDelay` filter would also be appropriate. See Section 11.4.44, "TimeDelay (delay) Filter" for more detail.

| | | | |
|---|---|---|---|
| Pre-configured filter name | | `delayInMs` | |
| Classname | | `com.continuent.tungsten.replicator.filter.TimeDelayMsFilter` | |
| Property prefix | | *`replicator.filter.delayInMs`* | |
| Stage compatibility | | remote-to-thl | |
| tpm Option compatibility | | `--repl-svc-thl-filters` [426] | |
| Data compatibility | | Any event | |
| Parameters | | | |
| Parameter | Type | Default | Description |
| *`delay`* | numeric | 60000 | Number of milliseconds to delay transaction processing row |

The `TimeDelayMsFilter` delays the application of transactions recorded in the THL. The delay can be used to allow point-in-time recovery of DML operations before the transaction has been applied to the Target, or where data may need to be audited or checked before transactions are committed.

### Note

For effective operation, Source and Targets should be synchronized using NTP or a similar protocol.

To enable the `TimeDelayMsFilter`, update the tungsten.ini configuration file to enable the filter. For example, to enable the delay for 120000 milliseconds:

```
shell> vi /etc/tungsten/tungsten.ini
[serviceName]
...
svc-applier-filters=delayInMs
property=replicator.filter.delayInMs.delay=120000
...

shel> tpm update
```

Time delay of transaction events should be performed with care, since the delay will prevent an Target from being up to date compared to the Source. In the event of a node failure, an up to date Target is required to ensure that data is safe.

## 11.4.46. `tosingledb.js` Filter

This filter updates the replicated information so that it goes to an explicit schema, as defined by the user. The filter can be used to combine multiple tables to a single schema.

| | | | |
|---|---|---|---|
| Pre-configured filter name | | `tosingledb` | |
| JavaScript Filter File | | `tungsten-replicator/support/filters-javascript/tosingledb.js` | |
| Property prefix | | *`replicator.filter.ansiquotes`* | |
| Stage compatibility | | `q-to-dbms` | |
| tpm Option compatibility | | `--svc-applier-filters` [424] | |
| Data compatibility | | Any event | |
| Parameters | | | |
| Parameter | Type | Default | Description |
| *`db`* | `string` | (none) | Database name into which to replicate all tables |
| *`skip`* | `string` | (none) | Comma-separated list of databases to be ignored |

A database can be optionally ignored through the `skip` parameter within the configuration:

```
--property=replicator.filter.tosingledb.db=centraldb \
--property=replicator.filter.tosingledb.skip=tungsten
```

The above configures all data to be written into `centraldb`, but skips the database `tungsten`.

Similar to other filters, the filter operates by explicitly changing the schema name to the configured schema, unless the skipped schema is in the event data. For example, at a statement level:

```
if(oldDb!=null && oldDb.compareTo(skip)!=0)
{
  d.setDefaultSchema(db);
}
```

## 11.4.47. `truncatetext.js` Filter

The `truncatetext` filter truncates a MySQL `BLOB` field.

| | | | |
|---|---|---|---|
| Pre-configured filter name | | `truncatetext` | |
| JavaScript Filter File | | `tungsten-replicator/support/filters-javascript/truncatetext.js` | |
| Property prefix | | *`replicator.filter.truncatetext`* | |
| Stage compatibility | | `binlog-to-q`, `q-to-dbms` | |
| tpm Option compatibility | | `--svc-extractor-filters` [425], `--svc-extractor-filters` [425] | |
| Data compatibility | | Row events | |
| Parameters | | | |
| Parameter | Type | Default | Description |
| *`length`* | `numeric` | (none) | Maximum size of truncated field (bytes) |

The length is determined by the *`length`* parameter in the properties:

```
replicator.filter.truncatetext=com.continuent.tungsten.replicator.filter.JavaScriptFilter
replicator.filter.truncatetext.script=${replicator.home.dir}/samples/extensions/javascript/truncatetext.js
replicator.filter.truncatetext.length=4000
```

Statement-based events are ignored, but row-based events are processed for each volume value, checking the column type, `isBlob()` method and then truncating the contents when they are identified as larger than the configured length. To confirm the type, it is compared against the Java class `com.continuent.tungsten.replicator.extractor.mysql.SerialBlob`, the class for a serialized `BLOB` value. These need to be processed differently as they are not exposed as a single variable.

```
if (value.getValue() instanceof com.continuent.tungsten.replicator.extractor.mysql.SerialBlob)
{
  blob = value.getValue();
  if (blob != null)
  {
    valueBytes = blob.getBytes(1, blob.length());
    if (blob.length() > truncateTo)
    {
      blob.truncate(truncateTo);
    }
  }
}
```

## 11.4.48. `zerodate2null.js` Filter

The `zerodate2null` filter looks complicated, but is very simple. It processes row data looking for date columns. If the corresponding value is zero within the column, the value is updated to NULL. This is required for MySQL to Oracle replication scenarios.

| | | | |
|---|---|---|---|
| Pre-configured filter name | | `zerodate2null` | |
| JavaScript Filter File | | `tungsten-replicator/support/filters-javascript/zerodate2null.js` | |
| Property prefix | | *`replicator.filter.zerodate2null`* | |
| Stage compatibility | | `q-to-dbms` | |
| tpm Option compatibility | | `--svc-applier-filters` [424] | |
| Data compatibility | | Row events | |
| Parameters | | | |
| Parameter | Type | Default | Description |

The filter works by examining the column specification using the `getColumnSpec()` method. Each column is then checked to see if the column type is a `DATE`, `DATETIME` or `TIMESTAMP` by looking the type ID using some stored values for the date type.

Because the column index and corresponding value index match, when the value is zero, the column value is explicitly set to NULL using the `setValueNull()` method.

```
for(j = 0; j < rowChanges.size(); j++)
{
  oneRowChange = rowChanges.get(j);
  columns = oneRowChange.getColumnSpec();
  columnValues = oneRowChange.getColumnValues();
  for (c = 0; c < columns.size(); c++)
  {
    columnSpec = columns.get(c);
    type = columnSpec.getType();
    if (type == TypesDATE || type == TypesTIMESTAMP)
    {
      for (row = 0; row < columnValues.size(); row++)
      {
        values = columnValues.get(row);
        value = values.get(c);

        if (value.getValue() == 0)
        {
          value.setValueNull()
        }
      }
    }
  }
}
```

# 11.5. Standard JSON Filter Configuration

A number of the filters that are included as part of Tungsten Cluster use a standardised form of configuration file that is designed to easy to use and familiar, while being flexible enough to support the needs of each filter. For the majority of filter configurations, the core focus of the configuration is based on a 'default' setting, and settings that are specific to a schema or table.

The JSON configuration follows this basic model. The following filters support the use of this JSON configuration file format:

- `convertstringfrommysql`
- `pkey`
- `skipeventbytype`

The basic format of the configuration is a JSON file that is split into two sections:

- A default section, which determines what will happen in the absence of a schema/table specific rule.
- A collection of schema and table specific entries that determine what happens for a specific schema/table combination.

Depending on the filter and use case, the information within both sections can then either be further divided into column-specific information, or the information may be configured as key/value pairs, or objects, to configure individual parts of the filter configuration.

For example, the following configuration file is from the `pkey` filter:

```
{
    "__default": {
        "IGNORE" : "pkey"
    },
    "test" : {
        "msg" : {
            "msg" : "pkey"
        }
    }
}
```

The above shows the the defaults section, and the schema/table specific section.

### Note

Depending on the filter, the default section may merely be a placeholder to indicate the format of the file. The `_defaults` should never be removed.

The sample shows a full schema name, table name, and then column name configuration.

By comparison, the sample below has only schema and table name information, with the configuration within that section being used to define the key/value pairs for specific operations as part of the `skipeventbytype` filter:

```
{
```

```
    "__default": {
        "INSERT" : "allow",
        "DELETE" : "allow",
        "UPDATE" : "allow"
    },
    "SCHEMA" : {
        "TABLE" : {
            "INSERT" : "allow",
            "DELETE" : "deny",
            "UPDATE" : "deny"
        }
    }
}
```

The selection and execution of the rules is determined by some specific rules, as detailed in Section 11.5.1, “Rule Handling and Processing” and Section 11.5.2, “Schema, Table, and Column Selection”.

## 11.5.1. Rule Handling and Processing

The processing of the rules and the selection of the tables and appropriate response and operation is configured through the combination of the default and schema/table settings according to explicit rules:

- If the incoming data matches the schema and table (and optionally column) according to the rules, use the configuration information in that section.
- If the schema/table is not specified or does not have explicit configuration, use the configuration within the `__defaults` section instead.

The default rule is always processed and followed if there is no match for an explicit schema, table, or column definition.

## 11.5.2. Schema, Table, and Column Selection

The format of the JSON configuraiton and the selection of the schema, table, and column information is in the form of nested structure of JSON objects. The schema first, then the table, then optionally the column within a nested JSON structure. For example:

```
"test" : {
    "msg" : {
        "id" : "pkey"
    }
}
```

In the above example:

- `test` is the schema name
- `msg` is the table name within the `test` schema
- `id` is the column name within the `test.msg` table

For different tables within the same schema, place another entry at the same level:

```
"test" : {
    "msg" : {
        "id" : "pkey"
    },
    "orders" : {
        "id" : "pkey"
    },
}
```

The above now handles the tables `msg` and `orders` within the `test` schema.

Wildcards are also supported, using the `*` operator. For example:

```
"orders" : {
    "*" : {
        "INSERT" : "allow",
        "DELETE" : "deny",
        "UPDATE" : "deny"
    }
}
```

Would match all tables within the `orders` schema. If multiple definitions exist, then the matching operates on the closest match first. For example:

```
"orders" : {
    "sales" : {
```

```
        "INSERT" : "deny",
        "DELETE" : "deny",
        "UPDATE" : "deny"
    },
    "*" : {
        "INSERT" : "allow",
        "DELETE" : "deny",
        "UPDATE" : "deny"
    }
}
```

In the above, if the schema/table combination `orders.sales` is seen, the rule for that is always used first as it is explicitly stated. Only tables that do not match the wildcards will use the wildcard entry. If neither an explicit schema/table or wildcard exist, the default is used.

# 11.6. JavaScript Filters

In addition to the supplied Java filters, Tungsten Replicator also includes support for custom script-based filters written in JavaScript and supported through the JavaScript filter. This filter provides a JavaScript environment that exposes the transaction information as it is processed internally through an object-based JavaScript API.

The JavaScript implementation is provided through the Rhino open-source implementation. Rhino provides a direct interface between the underlying Java classes used to implement the replicator code and a full JavaScript environment. This enables scripts to be developed that have access to the replicator constructs and data structures, and allows information to be updated, reformatted, combined, extracted and reconstructed.

At the simplest level, this allows for operations such as database renames and filtering. More complex solutions allow for modification of the individual data, such as removing nulls, bad dates, and duplication of information.

### Warning

If you previously implemented custom filters with older releases of Tungsten Replicator or with the now deprecated Open Source (OSS) release, you would have edited the `static-SERVICE.properties` file.

This is no longer a supported method of implementing custom filters, and doing so will break automated upgrades through tpm.

To enable custom filters, follow the process here: Section 11.6.2, “Installing Custom JavaScript Filters”

## 11.6.1. Writing JavaScript Filters

The JavaScript interface to the replicator enables filters to be written using standard JavaScript with a complete object-based interface to the internal Java objects and classes that make up the THL data.

For more information on the Rhino JavaScript implementation, see Rhino.

The basic structure of a JavaScript filter is as follows:

```
// Prepare the filter and setup structures

prepare()
{

}

// Perform the filter process; function is called for each event in the THL

filter(event)
{

// Get the array of DBMSData objects
    data = event.getData();

    // Iterate over the individual DBMSData objects
    for(i=0;i<data.size();i++)
    {
      // Get a single DBMSData object
      d = data.get(i);

      // Process a Statement Event; event type is identified by comparing the object class type

      if (d = instanceof com.continuent.tungsten.replicator.dbms.StatementData)
      {
        // Do statement processing
      }
      else if (d = instanceof com.continuent.tungsten.replicator.dbms.RowChangeData)
```

```
      {
        // Get an array of all the row changes
        rows = data.get(i).getRowChanges();

        // Iterate over row changes
        for(j=0;j<rows.size();j++)
        {
          // Get the single row change
          rowchange = rows.get(j);

          // Identify the row change type
          if (rowchange.getAction() == "INSERT")
          {
          }
          ....

      }
    }
  }
}
```

The following sections will examine the different data structures, functions, and information available when processing these individual events.

### 11.6.1.1. Implementable Functions

Each JavaScript filter must defined one or more functions that are used to operate the filter process. The `filter()` [492] function must be defined, as it contains the primary operation sequence for the defined filter. The function is supplied the event from the THL as the events are processed by the replicator.

In addition, two other JavaScript functions can optionally be defined that are executed before and after the filter process. Additional, user-specific, functions can be defined within the filter context to support the filter operations.

- *`prepare()`*

  The `prepare()` [492] function is called when the replicator is first started, and initializes the configured filter with any values that may be required during the filter process. These can include loading and identifying configuration values, creating lookup, exception or other reference tables and other internal JavaScript tables based on the configuration information, and reporting the generated configuration or operation for debugging.

- *`filter(event)`*

  The `filter()` [492] function is the main function that is called each time an event is loaded from the THL. The *`event`* is parsed as the only parameter to the function and is an object containing all the statement or row data for a given event.

- *`release()` [492]*

  The `release()` [492] function is called when the filter is deallocated and removed, typically during shutdown of the replicator, although it may also occur when a processing thread is restarted.

### 11.6.1.2. Getting Configuration Parameters

The JavaScript interface enables you to get two different sets of configuration properties, the filter specific properties, and the general replicator properties. The filter specific properties should be used configure and specify configuration information unique to that instance of the filter configuration. Since multiple filter configurations using the same filter definition can be created, using the filter-specific content is the simplest method for obtaining this information.

- Getting Filter Properties

  To obtain the properties configured for the filter within the static configuration file according to the context of the filter configuration, use the `filterProperties` class with the `getString()` method. For example, the `dbrename` filter uses two properties, dbsource and dbtarget to identify the database to be renamed and the new name. The definition for the filter within the configuration file might be:

  ```
  replicator.filter.jsdbrename=com.continuent.tungsten.replicator.filter.JavaScriptFilter
  replicator.filter.jsdbrename.script=${replicator.home.dir}/support/filters-javascript/dbrename.js
  replicator.filter.jsdbrename.dbsource=contacts
  replicator.filter.jsdbrename.dbtarget=nyc_contacts
  ```

  Within the JavaScript filter, they are retrieved using:

  ```
  sourceName = filterProperties.getString("dbsource");
  targetName = filterProperties.getString("dbtarget");
  ```

- Generic Replicator Properties

General properties can be retrieved using the `properties` class and the `getString()` method:

```
master = properties.getString("replicator.thl.remote_uri");
```

### 11.6.1.3. Logging Information and Exceptions

Information about the filtering process can be reported into the standard `trepsvc.log` file by using the `logger` object. This supports different methods according to the configured logging level:

- `logger.info()` — information level entry, used to indicate configuration, loading or progress.
- `logger.debug()` — information will be logged when debugging is enabled, used when showing progress during development.
- `logger.error()` — used to log an error that would cause a problem or replication to stop.

For example, to log an informational entry that includes data from the filter process:

```
logger.info("regexp: Translating string " + valueString.valueOf());
```

To raise an exception that causes replication to stop, a new `ReplicatorException` object must be created that contains the error message:

```
if(col == null)
{
  throw new com.continuent.tungsten.replicator.ReplicatorException(
    "dropcolumn.js: column name in " + schema + "." + table +
    " is undefined - is colnames filter enabled and is it before the dropcolumn filter?"
    );
}
```

The error string provided will be used as the error provided through trepctl, in addition to raising and exception and backtrace within the log.

### 11.6.1.4. Exposed Data Structures

Within the `filter()` [492] function that must be defined within the JavaScript filter, a single *`event`* object is supplied as the only argument. That event object contains all of the information about a single event as recorded within the THL as part of the replication process. Each event contains metadata information that can be used to identify or control the content, and individual statement and row data that contain the database changes.

The content of the information is a compound set of data that contains one or more further blocks of data changes, which in turn contains one or more blocks of SQL statements or row data. These blocks are defined using the Java objects that describe their internal format, and are exposed within the JavaScript wrapper as JavaScript objects, that can be parsed and manipulated.

At the top level, the Java object provided to the to the `filter()` [492] function as the *`event`* argument is `ReplDBMSEvent`. The `ReplDBMSEvent` class provides the core event information with additional management metadata such as the global transaction ID (seqno), latency of the event and sharding information.

That object contains one or more `DBMSData` objects. Each `DBMSData` object contains either a `StatementData` object (in the case of a statement based event), or a `RowChangeData` object (in the case of row-based events). For row-based events, there will be one or more `OneRowChange` [496] objects for each individual row that was changed.

When processing the event information, the data that is processed is live and should be updated in place. For example, when examining statement data, the statement needs only be updated in place, not re-submitted. Statements and rows can also be explicitly removed or added by deleting or extending the arrays that make up the objects.

A basic diagram of the structure is shown in the diagram below:

| | | | |
|---|---|---|---|
| `ReplDBMSEvent` | `DBMSData` | `StatementData` | |
| | `DBMSData` | `StatementData` | |
| | `DBMSData` | `RowChangeData` | `OneRowChange` [496] |
| | | | `OneRowChange` [496] |
| | | | ... |
| | | `StatementData` | |
| `ReplDBMSEvent` | `DBMSData` | `RowChangeData` | `OneRowChange` [496] |
| | | | `OneRowChange` [496] |
| | | | ... |

A single event can contain both statement and row change information within the list of individual `DBMSData` events. An event or

#### 11.6.1.4.1. `ReplDBMSEvent` Objects

The base object from which all of the data about replication can be obtained is the `ReplDBMSEvent` class. The class contains all of the information about each event, including the global transaction ID and statement or row data.

The interface to the underlying information is through a series of methods that provide the embedded information or data structures, described in the table below.

| Method | Description |
|---|---|
| `getAppliedLatency()` | Returns the latency of the embedded event. See Section E.2.8, "Terminology: Fields *appliedLatency*" |
| `getData()` | Returns an array of the `DBMSData` objects within the event |
| `getDBMSEvent()` | Returns the original `DBMSEvent` object |
| `getEpochNumber()` | Get the Epoch number of the stored event. See THL EPOCH# [550] |
| `getEventId()` | Returns the native event ID. See THL EVENTID [550] |
| `getExtractedTstamp()` | Returns the timestamp of the event. |
| `getFragno()` | Returns the fragment ID. See THL SEQNO [549] |
| `getLastFrag()` | Returns true if the fragment is the last fragment in the event. |
| `getSeqno()` | Returns the native sequence number. See THL SEQNO [549] |
| `getShardId()` | Returns the shard ID for the event. |
| `getSourceId()` | Returns the source ID of the event. See THL SOURCEID [550] |
| `setShardId()` | Sets the shard ID for the event, which can be used by the filter to set the shard. |

The primary method used is `getData()`, which returns an array of the individual `DBMSData` objects contain in the event:

```
function filter(event)
{
  data = event.getData();

if(data != null)
  {

    for (i = 0; i < data.size(); i++)
    {
        change = data.get(i);
...
```

Access to the underlying array structure uses the `get()` method to request individual objects from the array. The `size()` method returns the length of the array.

Removing or Adding Data Changes

Individual `DBMSData` objects can be removed from the replication stream by using the `remove()` method, supplying the index of the object to remove:

```
data.remove(1);
```

The `add()` method can be used to add new data changes into the stream. For example, data can be duplicated across tables by creating and adding a new version of the event, for example:

```
if(d.getDefaultSchema() != null &&
   d.getDefaultSchema().compareTo(sourceName)==0)
{
  newStatement = new
     com.continuent.tungsten.replicator.dbms.StatementData(d.getQuery(),
                                                           null,
                                                           targetName);
  data.add(data.size(),newStatement);
}
```

The above code looks for statements within the `sourceName` schema and creates a copy of each statement into the `targetName` schema.

The first argument to `add()` is the index position to add the statement. Zero (0) indicates before any existing changes, while using `size()` on the array effectively adds the new statement change at the end of the array.

Updating the Shard ID

The `setShardId()` method can also be used to set the shard ID within an event. This can be used in filters where the shard ID is updated by examining the schema or table being updated within the embedded SQL or row data. An example of this is provided in Section 11.4.42, "`shardbytable.js` Filter".

### 11.6.1.4.2. `DBMSData` Objects

The `DBMSData` object provides encapsulation of either the SQL or row change data within the THL. The class provides no methods for interacting with the content, instead, the real object should be identified and processed accordingly. Using the JavaScript `instanceof` operator the underlying type can be determined:

```
if (d != null &&
    d instanceof com.continuent.tungsten.replicator.dbms.StatementData)
{
   // Process Statement data
}
else if (d != null &&
    d instanceof com.continuent.tungsten.replicator.dbms.RowChangeData)
{
   // Process Row data
}
```

Note the use of the full object class for the different `DBMSData` types.

For information on processing `StatementData`, see Section 11.6.1.4.3, "`StatementData` Objects". For row data, see Section 11.6.1.4.4, "`RowChangeData` Objects".

### 11.6.1.4.3. `StatementData` Objects

The `StatementData` class contains information about data that has been replicated as an SQL statement, as opposed to information that is replicated as row-based data.

Processing and filtering statement information relies on editing the original SQL query statement, or the metadata recorded with it in the THL, such as the schema name or character set. Care should be taken when modifying SQL statement data to ensure that you are modifying the right part of the original statement. For example, a search and replace on an SQL statement should be made with care to ensure that embedded data is not altered by the process.

The key methods used for interacting with a `StatementData` object are listed below:

| Method | Description |
|---|---|
| `getQuery()` | Returns the SQL statement |
| `setQuery()` | Updates the SQL statement |
| `appendToQuery()` | Appends a string to an existing query |
| `getDefaultSchema()` | Returns the default schema in which the statement was executed. The schema may be null for explicit or multi-schema queries. |
| `setDefaultSchema()` | Set the default schema for the SQL statement |
| `getTimestamp()` | Gets the timestamp of the query. This is required if data must be applied with a relative value by combining the timestamp with the relative value |

Updating the SQL

The primary method of processing statement based data is to load and identify the original SQL statement (using `getQuery()`, update or modify the SQL statement string, and then update the statement within the THL again using `setQuery()`. For example:

```
sqlOriginal = d.getQuery();
sqlNew = sqlOriginal.replaceAll('NOTEPAD','notepad');
d.setQuery(sqlNew);
```

The above replaces the uppercase 'NOTEPAD' with a lowercase version in the query before updating the stored query in the object.

Changing the Schema Name

Some schema and other information is also provided in this structure. For example, the schema name is provided within the statement data and can be explicitly updated. In the example below, the schema "products" is updated to "nyc_products":

```
if (change.getDefaultSchema().compareTo("products") == 0)
{
  change.setDefaultSchema("nyc_products");
}
```

A similar operation should be performed for any row-based changes. A more complete example can be found in Section 11.4.8, "`dbrename.js` Filter".

### 11.6.1.4.4. RowChangeData Objects

RowChangeData is information that has been written into the THL in row format, and therefore consists of rows of individual data divided into the individual columns that make up each row-based change. Processing of these individual changes must be performed one row at a time using the list of OneRowChange [496] objects provided.

The following methods are supported for the RowChangeData object:

| Method | Description |
|---|---|
| appendOneRowChange(rowChange) | Appends a single row change to the event, using the supplied OneRowChange [496] object. |
| getRowChanges() | Returns an array list of all the changes as OneRowChange [496] objects. |
| setRowChanges(rowChanges) | Sets the row changes within the event using the supplied list of OneRowChange objects. |

For example, a typical row-based process will operate as follows:

```
if (d != null && d instanceof com.continuent.tungsten.replicator.dbms.RowChangeData)
 {
   rowChanges = d.getRowChanges();

   for(j = 0; j < rowChanges.size(); j++)
   {
     oneRowChange = rowChanges.get(j);
     // Do row filter
```

The OneRowChange [496] object contains the changes for just one row within the event. The class contains the information about the tables, field names and field values. The following methods are supported:

| Method | Description |
|---|---|
| getAction() | Returns the row action type, i.e. whether the row change is an INSERT, UPDATE or DELETE |
| getColumnSpec() | Returns the specification of each column within the row change |
| getColumnValues() | Returns the value of each column within the row change |
| getSchemaName() | Gets the schema name of the row change |
| getTableName() | Gets the table name of the row change |
| setColumnSpec() | Sets the column specification using an array of column specifications |
| setColumnValues() | Sets the column values |
| setSchemaName() | Sets the schema name |
| setTableName() | Sets the table name |

Changing Schema or Table Names

The schema, table and column names are exposed at different levels within the OneRowChange [496] object. Updating the schema name can be achieved by getting and setting the name through the getSchemaName() and setSchemaName() methods. For example, to add a prefix to a schema name:

```
rowchange.setSchemaName('prefix_' + rowchange.getSchemaName());
```

To update a table name, the getTableName() and setTableName() can be used in the same manner:

```
oneRowChange.setTableName('prefix_' + oneRowChange.getTableName());
```

Getting Action Types

Row operations are categorised according to the action of the row change, i.e. whether the change was an insert, update or delete operation. This information can be extracted from each row change by using the getAction() method:

```
action = oneRowChange.getAction();
```

The action information is returned as a string, i.e. INSERT, UPDATE, or DELETE. This enables information to be filtered according to the changes; for example by selectively modifying or altering events.

For example, DELETE events could be removed from the list of row changes:

```
for(j=0;j<rowChanges.size();j++)
{
  oneRowChange = rowChanges.get(j);
```

```
  if (oneRowChange.actionType == 'DELETE')
  {
    rowChanges.remove(j);
    j--;
  }
}
```

The `j--` is required because as each row change is removed, the size of the array changes and our current index within the array needs to be explicitly modified.

Extracting Column Definitions

To extract the row data, the `getColumnValues()` method returns the an array containing the value of each column in the row change. Obtaining the column specification information using `getColumnSpec()` returns a corresponding specification of each corresponding column. The column data can be used to obtain the column type information

To change column names or values, first the column information should be identified. The column information in each row change should be retrieved and/or updated. The `getColumnSpec()` returns the column specification of the row change. The information is returned as an array of the individual columns and their specification:

```
columns = oneRowChange.getColumnSpec();
```

For each column specification a `ColumnSpec` object is returned, which supports the following methods:

| Method | Description |
|---|---|
| `getIndex()` | Gets the index of the column within the row change |
| `getLength()` | Gets the length of the column |
| `getName()` | Returns the column name if available |
| `getType()` | Gets the type number of the column |
| `getTypeDescription()` | |
| `isBlob()` | Returns true if the column is a blob |
| `isNotNull()` | Returns true if the column is configured as `NOT NULL` |
| `isUnsigned()` | Returns true if the column is unsigned. |
| `setBlob()` | Set the column blob specification |
| `setIndex()` | Set the column index order |
| `setLength()` | Returns the column length |
| `setName()` | Set the column name |
| `setNotNull()` | Set whether the column is configured as `NOT NULL` |
| `setSigned()` | Set whether the column data is signed |
| `setType()` | Set the column type |
| `setTypeDescription()` | Set the column type description |

To identify the column type, use the `getType()` method which returns an integer matching the underlying data type. There are no predefined types, but common values include:

| Type | Value | Notes |
|---|---|---|
| `INT` | 4 | |
| `CHAR` or `VARCHAR` | 12 | |
| `TEXT` or `BLOB` | 2004 | Use `isBlob()` to identify if the column is a blob or not |
| `TIME` | 92 | |
| `DATE` | 91 | |
| `DATETIME` or `TIMESTAMP` | 92 | |
| `DOUBLE` | 8 | |

Other information about the column, such as the length, and value types (unsigned, null, etc.) can be determined using the other functions against the column specification.

Extracting Row Data

The `getColumnValues()` method returns an array that corresponds to the information returned by the `getColumnSpec()` method. That is, the method returns a complementary array of the row change values, one element for each row, where each row is itself a further array of each column:

```
values = oneRowChange.getColumnValues();
```

This means that index 0 of the array from `getColumnSpec()` refers to the same column as index 0 of the array for a single row from `getColumnValues()`.

| **getColumnSpec()** | msgid | message | msgdate |
|---|---|---|---|
| `getColumnValues()` | | | |
| `[0]` | 1 | Hello New York! | Thursday, June 13, 2013 |
| `[1]` | 2 | Hello San Francisco! | Thursday, June 13, 2013 |
| `[2]` | 3 | Hello Chicago! | Thursday, June 13, 2013 |

This enables the script to identify the column type by the index, and then the corresponding value update using the same index. In the above example, the `message` field will always be index 1 within the corresponding values.

Each value object supports the following methods:

| Method | Description |
|---|---|
| `getValue()` | Get the current column value |
| `setValue()` | Set the column value to the supplied value |
| `setValueNull()` | Set the column value to NULL |

For example, within the `zerodate2null` sample, dates with a zero value are set to NULL using the following code:

```
columns = oneRowChange.getColumnSpec();
columnValues = oneRowChange.getColumnValues();
for (c = 0; c < columns.size(); c++)
{
  columnSpec = columns.get(c);
  type = columnSpec.getType();

  if (type == TypesDATE || type == TypesTIMESTAMP)
  {
    for (row = 0; row < columnValues.size(); row++)
    {
      values = columnValues.get(row);
      value = values.get(c);

      if (value.getValue() == 0)
      {
        value.setValueNull()
      }
    }
  }
}
```

In the above example, the column specification is retrieved to determine which columns are date types. Then the list of embedded row values is extracted, and iterates over each row, setting the value for a date that is zero (0) to be NULL using the `setValueNull()` method.

An alternative would be to update to an explicit value using the `setValue()` method.

## 11.6.2. Installing Custom JavaScript Filters

Once you have written your JavaScript filter, and ready to install it you need to follow the steps below. This will allow you to configure and apply the filter to your installation using the standard tpm procedure.

For this example, we will assume your new JavaScript file is called number2binary.js, and the filter has two additional boolean configuration properties 'roundup' and 'debug'

### 11.6.2.1. Step 1: Copy JavaScript files

By default, the software package will be contained in `/opt/continuent/software/tungsten-replicator-7.1.2-81` Adjust the path in the examples accordingly if your environment differs.

The JavaScript file for your new filter(s) need copying to the following location:

```
/opt/continuent/software/tungsten-replicator-7.1.2-81/tungsten-replicator/samples/extensions/javascript
```

### 11.6.2.2. Step 2: Create Template Files

You need to create a template file which contains the location of the JavaScript file and the additional configuration properties with the appropriate default values.

Create a file called number2binary.tpl that contains the following:

```
replicator.filter.number2binary=com.continuent.tungsten.replicator.filter.JavaScriptFilter
replicator.filter.number2binary.script=${replicator.home.dir}/samples/extensions/javascript/number2binary.js
replicator.filter.number2binary.roundup=true
replicator.filter.number2binary.debug=false
```

This tpl file needs to be copied into the following directory:

```
/opt/continuent/software/tungsten-replicator-7.1.2-81/tungsten-replicator/samples/conf/filters/default
```

### 11.6.2.3. Step 3: (Optional) Copy json files

If your filter uses json files to load configuration data, this needs to be copied into the `/opt/continuent/share` directory and also included in the tpl file created in Step 2. An example is as follows:

```
replicator.filter.{FILTERNAME}.definitionsFile=/opt/continuent/share/{FILTERNAME}.json
```

### 11.6.2.4. Step 4: Update Configuration

Now that all the files are in place you can include the custom filter in your configuration.

Any properties set with a default value in the tpl file, only need including if you wish to overwrite the default value

The following examples show how you can now include this in your tpm configuration:

For ini installations add the following to your tungsten.ini

```
svc-extractor-filters={existing filter definitions},number2binary
property=replicator.filter.number2binary.roundup=false
property=replicator.filter.number2binary.debug=true
```

For staging installations

```
shell> cd {staging-dir}

shell> tools/tpm configure SERVICENAME
{other-configuration-values-as-requires} \
--svc-extractor-filters={existing filter definitions},number2binary \
--property=replicator.filter.number2binary.roundup=false \
--property=replicator.filter.number2binary.debug=true

shell> tools/tpm install
```

In the above examples we used the `svc-extractor-filters` [425] property for the extractor replicator. If you are applying your custom filters to your applier, then use `svc-applier-filters` [424] instead

Your custom filters are now installed in a clean and easy to manage process allowing you to use tpm for all future update processes

If there is a problem with the JavaScript filter during restart, the replicator will be placed into the `OFFLINE` [195] state and the reason for the error will be provided within the replicator `trepsvc.log` log.

# Chapter 12. Performance and Tuning

Whilst in most cases, very little tuning is required within the replicator, there may be times when you need to tweak the replicator to better suit your workloads. The sections in this chapter cover the most common tuning tasks you may wish to consider.

## 12.1. Block Commit

The replicator commits changes read from the THL and commits these changes in Replicas during the applier stage according to the block commit size or interval. These replace the single *replicator.global.buffer.size* parameter that controls the size of the buffers used within each stage of the replicator.

When applying transactions to the database, the decision to commit a block of transactions is controlled by two parameters:

- When the event count reaches the specified event limit (set by `--svc-applier-block-commit-size` [424])
- When the commit timer reaches the specified commit interval (set by `--svc-applier-block-commit-interval` [424])

The default operation is for block commits to take place based on the transaction count. Commits by the timer are disabled. The default block commit size is 10 transactions from the incoming stream of THL data; the block commit interval is zero (0), which indicates that the interval is disabled.

When both parameters are configured, block commit occurs when either value limit is reached. For example,if the event count is set to 10 and the commit interval to 50s, events will be committed by the applier either when the event count hits 10 or every 50 seconds, whichever is reached first. This means, for example, that even if only one transaction exists, when the 50 seconds is up, that single transaction will be applied.

In addition, the execution of implied commits during specific events within the replicator can also be controlled to prevent fragmented block commits by using the *replicator.stage.q-to-dbms.blockCommitPolicy* property. This property can have either of the following values:

- `strict` — Commit block on service name changes, multiple fragments in a transaction, or unsafe_for_block_commit. This is the default setting.
- `lax` — Don't commit in any of these cases.

The block commit size can be controlled using the `--repl-svc-applier-block-commit-size` [424] option to tpm, or through the `blockCommitRowCount`.

The block commit interval can be controlled using the `--repl-svc-applier-block-commit-interval` [424] option to tpm, or through the `blockCommitInterval`. If only a number is supplied, it is used as the interval in milliseconds. Suffix of s, m, h, and d for seconds, minutes, hours and days are also supported.

```
shell> ./tools/tpm update alpha \
    --repl-svc-applier-block-commit-size=20 \
    --repl-svc-applier-block-commit-interval=100s
```

**Note**

The block commit parameters are supported only in applier stages; they have no effect in other stages.

Modification of the block commit interval should be made only when the commit window needs to be altered. The setting can be particularly useful in heterogeneous deployments where the nature and behaviour of the target database is different to that of the source extractor.

For example, when replicating to Oracle, reducing the number of transactions within commits reduces the locks and overheads:

```
shell> ./tools/tpm update alpha \
    --repl-svc-applier-block-commit-interval=500
```

This would apply two commits every second, regardless of the block commit size.

When replicating to a data warehouse engine, particularly when using batch loading, such as Redshift, Vertica and Hadoop, larger block commit sizes and intervals may improve performance during the batch loading process:

```
shell> ./tools/tpm update alpha \
    --repl-svc-applier-block-commit-size=100000 \
    --repl-svc-applier-block-commit-interval=60s
```

This sets a large block commit size and interval enabling large batch loading.

### 12.1.1. Monitoring Block Commit Status

The block commit status can be monitored using the trepctl status -name tasks command. This outputs the *lastCommittedBlockSize* and *lastCommittedBlockTime* values which indicate the size and interval (in seconds) of the last block commit.

```
shell> trepctl status -name tasks
Processing status command (tasks)...
...
NAME                     VALUE
----                     -----
appliedLastEventId    : mysql-bin.000015:0000000000001117;0
appliedLastSeqno      : 5271
appliedLatency        : 4656.231
applyTime             : 0.066
averageBlockSize      : 0.500
cancelled             : false
commits               : 10
currentBlockSize      : 0
currentLastEventId    : mysql-bin.000015:0000000000001117;0
currentLastFragno     : 0
currentLastSeqno      : 5271
eventCount            : 5
extractTime           : 0.394
filterTime            : 0.017
lastCommittedBlockSize: 1
lastCommittedBlockTime: 0.033
otherTime             : 0.001
stage                 : q-to-dbms
state                 : extract
taskId                : 0
Finished status command (tasks)...
```

## 12.2. Improving Network Performance

The performance of the network can be critical when replicating data. The information transferred over the network contains the full content of the THL in addition to a small protocol overhead. Improving your network performance can have a significant impact on the overall performance of the replication process.

The following network parameters should be configured within your `/etc/sysctl.conf` and can safely applied to all the hosts within your cluster deployments:

```
# Increase size of file handles and inode cache
fs.file-max = 2097152

# tells the kernel how many TCP sockets that are not attached to any
# user file handle to maintain. In case this number is exceeded,
# orphaned connections are immediately reset and a warning is printed.
net.ipv4.tcp_max_orphans = 60000

# Do not cache metrics on closing connections
net.ipv4.tcp_no_metrics_save = 1

# Turn on window scaling which can enlarge the transfer window:
net.ipv4.tcp_window_scaling = 1

# Enable timestamps as defined in RFC1323:
net.ipv4.tcp_timestamps = 1

# Enable select acknowledgments:
net.ipv4.tcp_sack = 1

# Maximum number of remembered connection requests, which did not yet
# receive an acknowledgment from connecting client.
net.ipv4.tcp_max_syn_backlog = 10240

# recommended default congestion control is htcp
net.ipv4.tcp_congestion_control=htcp

# recommended for hosts with jumbo frames enabled
net.ipv4.tcp_mtu_probing=1

# Number of times SYNACKs for passive TCP connection.
net.ipv4.tcp_synack_retries = 2

# Allowed local port range
net.ipv4.ip_local_port_range = 1024 65535

# Protect Against TCP Time-Wait
net.ipv4.tcp_rfc1337 = 1

# Decrease the time default value for tcp_fin_timeout connection
net.ipv4.tcp_fin_timeout = 15

# Increase number of incoming connections
# somaxconn defines the number of request_sock structures
# allocated per each listen call. The
```

```
# queue is persistent through the life of the listen socket.
net.core.somaxconn = 1024

# Increase number of incoming connections backlog queue
# Sets the maximum number of packets, queued on the INPUT
# side, when the interface receives packets faster than
# kernel can process them.
net.core.netdev_max_backlog = 65536

# Increase the maximum amount of option memory buffers
net.core.optmem_max = 25165824

# Increase the maximum total buffer-space allocatable
# This is measured in units of pages (4096 bytes)
net.ipv4.tcp_mem = 65536 131072 262144
net.ipv4.udp_mem = 65536 131072 262144

### Set the max OS send buffer size (wmem) and receive buffer
# size (rmem) to 12 MB for queues on all protocols. In other
# words set the amount of memory that is allocated for each
# TCP socket when it is opened or created while transferring files

# Default Socket Receive Buffer
net.core.rmem_default = 25165824

# Maximum Socket Receive Buffer
net.core.rmem_max = 25165824

# Increase the read-buffer space allocatable (minimum size,
# initial size, and maximum size in bytes)
net.ipv4.tcp_rmem = 20480 12582912 25165824
net.ipv4.udp_rmem_min = 16384

# Default Socket Send Buffer
net.core.wmem_default = 25165824

# Maximum Socket Send Buffer
net.core.wmem_max = 25165824

# Increase the write-buffer-space allocatable
net.ipv4.tcp_wmem = 20480 12582912 25165824
net.ipv4.udp_wmem_min = 16384

# Increase the tcp-time-wait buckets pool size to prevent simple DOS attacks
net.ipv4.tcp_max_tw_buckets = 1440000
# net.ipv4.tcp_tw_recycle = 1
net.ipv4.tcp_tw_reuse = 1
```

## 12.3. Tungsten Replicator Block Commit and Memory Usage

Replicators are implemented as Java processes, which use two types of memory: stack space, which is allocated per running thread and holds objects that are allocated within individual execution stack frames, and heap memory, which is where objects that persist across individual method calls live. Stack space is rarely a problem for Tungsten as replicators rarely run more than 200 threads and use limited recursion. The Java defaults are almost always sufficient. Heap memory on the other hand runs out if the replicator has too many transactions in memory at once. This results in the dreaded Java OutOfMemory exception, which causes the replicator to stop operating. When this happens you need to look at tuning the replicator memory size.

To understand replicator memory usage, we need to look into how replicators work internally. Replicators use a "pipeline" model of execution that streams transactions through 1 or more concurrently executing stages. As you can can see from the attached diagram, an Applier pipeline might have a stage to read transactions from the Extractor and put them in the THL, a stage to read them back out of the THL into an in-memory queue, and a stage to apply those transactions to the Target. This model ensures high performance as the stages work independently. This streaming model is quite efficient and normally permits Tungsten to transfer even exceedingly large transactions, as the replicator breaks them up into smaller pieces called transaction fragments.

The pipeline model has consequences for memory management. First of all, replicators are doing many things at one, hence need enough memory to hold all current objects. Second, the replicator works fastest if the in-memory queues between stages are large enough that they do not ever become empty. This keeps delays in upstream processing from delaying things at the end of the pipeline. Also, it allows replicators to make use of block commit. Block commit is an important performance optimization in which stages try to commit many transactions at once on Targets to amortize the cost of commit. In block commit the end stage continues to commit transactions until it either runs out of work (i.e., the upstream queue becomes empty) or it hits the block commit limit. Larger upstream queues help keep the end stage from running out of work, hence increase efficiency.

Bearing this in mind, we can alter replicator behavior in a number of ways to make it use less memory or to handle larger amounts of traffic without getting a Java OutOfMemory error. You should look at each of these when tuning memory:

- Property wrapper.java.memory in file `wrapper.conf`. This controls the amount of heap memory available to replicators. 1024 MB is the minimum setting for most replicators. Busy replicators, those that have multiple services, or replicators that use parallel apply should consider

using 2048 MB instead. If you get a Java OutOfMemory exception, you should first try raising the current setting to a higher value. This is usually enough to get past most memory-related problems. You can set this at installation time as the `--repl-java-mem-size` [410] parameter.

- Property *`replicator.global.buffer.size`* in the replicator properties file. This controls two things, the size of in-memory queues in the replicator as well as the block commit size. If you still have problems after increasing the heap size, try reducing this value. It reduces the number of objects simultaneously stored on the Java heap. A value of 2 is a good setting to try to get around temporary problems. This can be set at installation time as the `--repl-buffer-size` [398] parameter.
- Property replicator.stage.q-to-dbms.blockCommitRowCount in the replicator properties file. This parameter sets the block commit count in the final stage in the Applier pipeline. If you reduce the global buffer size, it is a good idea to set this to a fixed size, such as 10, to avoid reducing the block commit effect too much. Very low block commit values in this stage can cut update rates on Targets by 50% or more in some cases. This is available at installation time as the `--repl-svc-applier-block-commit-size` [424] parameter.
- Property replicator.extractor.dbms.transaction_frag_size in the `replicator.properties` file. This parameter controls the size of fragments for long transactions. Tungsten automatically breaks up long transactions into fragments. This parameter controls the number of bytes of binlog per transaction fragment. You can try making this value smaller to reduce overall memory usage if many transactions are simultaneously present. Normally however this value has minimal impact.

Finally, it is worth mentioning that the main cause of out-of-memory conditions in replicators is large transactions. In particular, Tungsten cannot fragment individual statements or row changes, so changes to very large column values can also result in OutOfMemory conditions. For now the best approach is to raise memory, as described above, and change your application to avoid such transactions.

The replicator commits changes read from the THL and commits these changes in Targets during the applier stage according to the block commit size or interval. These replace the single *`replicator.global.buffer.size`* parameter that controls the size of the buffers used within each stage of the replicator.

When applying transactions to the database, the decision to commit a block of transactions is controlled by two parameters:

- When the event count reaches the specified event limit (set by `blockCommitRowCount`)
- When the commit timer reaches the specified commit interval (set by `blockCommitInterval`)

The default operation is for block commits to take place based on the transaction count. Commits by the timer are disabled. The default block commit size is 10 transactions from the incoming stream of THL data; the block commit interval is zero (0), which indicates that the interval is disabled.

When both parameters are configured, block commit occurs when either value limit is reached. For example,if the event count is set to 10 and the commit interval to 50s, events will be committed by the applier either when the event count hits 10 or every 50 seconds, whichever is reached first. This means, for example, that even if only one transaction exists, when the 50 seconds is up, that single transaction will be applied.

In addition, the execution of implied commits during specific events within the replicator can also be controlled to prevent fragmented block commits by using the *`replicator.stage.q-to-dbms.blockCommitPolicy`* property. This property can have either of the following values:

- *`strict`* — Commit block on service name changes, multiple fragments in a transaction, or unsafe_for_block_commit. This is the default setting.
- *`lax`* — Don't commit in any of these cases.

The block commit size can be controlled using the `--repl-svc-applier-block-commit-size` [424] option to tpm, or through the `blockCommitRowCount`.

The block commit interval can be controlled using the `--repl-svc-applier-block-commit-interval` [424] option to tpm, or through the `blockCommitInterval`. If only a number is supplied, it is used as the interval in milliseconds. Suffix of s, m, h, and d for seconds, minutes, hours and days are also supported.

```
shell> ./tools/tpm update alpha \
    --repl-svc-applier-block-commit-size=20 \
    --repl-svc-applier-block-commit-interval=100s
```

### Note

The block commit parameters are supported only in applier stages; they have no effect in other stages.

Modification of the block commit interval should be made only when the commit window needs to be altered. The setting can be particularly useful in heterogeneous deployments where the nature and behaviour of the target database is different to that of the source extractor.

For example, when replicating to Oracle, reducing the number of transactions within commits reduces the locks and overheads:

```
shell> ./tools/tpm update alpha \
    --repl-svc-applier-block-commit-interval=500
```

This would apply two commits every second, regardless of the block commit size.

When replicating to a data warehouse engine, particularly when using batch loading, such as Redshift, Vertica and Hadoop, larger block commit sizes and intervals may improve performance during the batch loading process:

```
shell> ./tools/tpm update alpha \
    --repl-svc-applier-block-commit-size=100000 \
    --repl-svc-applier-block-commit-interval=60s
```

This sets a large block commit size and interval enabling large batch loading.

# Appendix A. Release Notes

## A.1. Tungsten Replicator 7.1.2 GA (3 Apr 2024)

Version End of Life. Not Yet Set

Release 7.1.2 contains a number of key bug fixes and improvements.

### Note

v7.1.2 was re-released as build 81 on 13 May 2024 to fix a critical bug that exists in the original build 42 release (CT-2284).

Behavior Changes

The following changes have been made to Tungsten Replicator and may affect existing scripts and integration tools. Any scripts or environment which make use of these tools should check and update for the new configuration:

- Installation and Deployment
  - The systemd startup script will now have a dependency on the time-sync service (if available) being started before tungsten components.

    *Issues*: CT-2257

Improvements, new features and functionality

- Installation and Deployment
  - A new INI option has been added `rest-api-admin-password` which acts as an alias for the existing `rest-api-admin-pass` option.

    *Issues*: CT-2246
- Command-line Tools
  - A new command, tungsten_mysql_ssl_setup, has been introduced to create all needed security files for the MySQL database server. tungsten_mysql_ssl_setup will act as a direct replacement for the mysql_ssl_rsa_setup command which is not included with either Percona Server or MariaDB. This command will now be called by tpm cert gen mysqlcerts instead of mysql_ssl_rsa_setup.

    *Issues*: CT-2188
  - The tpm diag command now captures the output of sestatus and getenforce.

    *Issues*: CT-2243
- Filters
  - A new filter, binarystringconversionfilter, was added to provide a way to convert varchar data from one charset into another.

    *Issues*: CT-981

Bug Fixes

- Heterogeneous Replication
  - Fixes a bug in heterogeneous deployments that prevented CHAR and BINARY data types replicating correctly.

    *Issues*: CT-2258
  - Fixed an issue when using the ConvertStringFromMySQLFilter in heterogeneous deployments, that could show an error while processing an INSERT event after the PrimaryKeyFilter added key specifications to it.

    *Issues*: CT-2260
- Core Replicator
  - Event extraction can sometimes generate an empty fragment (for a multi fragment transaction). This in turn can lead to bad shard detection. While empty fragment is expected, shard detection will now use the last non-empty fragment.

    #### Note

    This bug was detected in v7.1.2 build 42, and fixed in v7.1.2 build 81

*Issues*: CT-2284

## A.2. Tungsten Replicator 7.1.1 GA (15 Dec 2023)

Version End of Life. Not Yet Set

Release 7.1.1 contains a number of key bug fixess and improvements.

Behavior Changes

The following changes have been made to Tungsten Replicator and may affect existing scripts and integration tools. Any scripts or environment which make use of these tools should check and update for the new configuration:

- Command-line Tools
  - The tpm diag command now collects `ps -aux` output in addition to `ps -efl` when possible.

    *Issues*: CT-2119

Bug Fixes

- Installation and Deployment
  - Installations will now succeed on hosts running Ruby 3.x

    *Issues*: CT-2196
- Command-line Tools
  - The tpm report command now correctly displays security info tpm options as blank when none exists.

    *Issues*: CT-2176
  - The tpm update command no longer fails to display foreign-owned dot-files and directories.

    *Issues*: CT-2190
  - The tpm mysql command now properly accepts the `-e` argument as a shortcut for the `--execute` mysql cli client argument.

    *Issues*: CT-2217
  - Fixes a regression in tpm cert that prevented `BASE_DIR` in `tungsten.env` from being used properly.

    *Issues*: CT-2237

## A.3. Tungsten Replicator 7.1.0 GA (16 Aug 2023)

Version End of Life. Not Yet Set

Release 7.1.0 is the next major v7 release containing a number of important bug fixes and key new features.

Behavior Changes

The following changes have been made to Tungsten Replicator and may affect existing scripts and integration tools. Any scripts or environment which make use of these tools should check and update for the new configuration:

- Command-line Tools
  - The tpm copy-keys command has been renamed to tpm copy and a new command has been created tpm cert copy with the same functionality.

    *Issues*: CT-2186

Improvements, new features and functionality

- Behavior Changes
  - Additional logging will now be added to the replicator logs during switchover/failover operations to enable better debugging in the event of issues.

    *Issues*: CT-1448

- Installation and Deployment
  - Running tpm uninstall will now save all of the Tungsten database tracking schemas for later use. There is also a new tpm keep command, which allows the tracking schemas to be saved to disk at any time in multiple formats (.json, .dmp and .cmd)

    *Issues*: CT-2131
  - Added tpm flag `deploy-systemd` as a more meaningful alias to `install`

    *Issues*: CT-2152
- Command-line Tools
  - Added a new option `--preserve-schema` to the tpm uninstall command in order to leave the tracking schema in the database.

    *Issues*: CT-561
  - A new command tpm cert has been added to aid in the creation, rotation and management of certificates for all areas of Tungsten.

    Known limitations: Percona 5.6, and all versions of MariaDB do not provide the mysql_ssl_rsa_setup command required by tpm cert gen mysqlcerts.

    *Issues*: CT-2085
  - The tpm diag command now supports the `--skipsudo` and `--nosudo` arguments to prevent operations from using the sudo command. Using this option may result in tpm diag skipping/failing various gathers due to a lack of access.

    *Issues*: CT-2146
- PostgreSQL Applier
  - It is now possible to configure the PostgreSQL applier in Batch mode.

    *Issues*: CT-2031
  - Added a new ddlscan template, `ddl-mysql-postgres-staging.vm`, to allow generation of staging table DDL for the PostgreSQL batch applier.

    *Issues*: CT-2136
- S3 CSV Applier
  - A new applier is now available that replicates to csv files on S3 storage.

    For more information, see S3 Applier Docuemntation

    *Issues*: CT-606
- Core Replicator
  - Upgraded JGroups library to 4.2.22

    *Issues*: CT-2011
- Filters
  - A new `shardbyrules` filter has been added that will allow rule based sharding of replication based on user confgurable rules that would allow sharding at table level, whereas previoulsy sharding would only be handled at schema level.

    For more information, see shardbyrules Filter Docuemntation

    *Issues*: CT-2164
- Monitoring
  - Prometheus exporters now provide the ssl cert expiration date as an epoch value in addition to the label.

    *Issues*: CT-2099
  - Prometheus libraries have been upgraded from version 0.8.1 to 0.16.0

    *Issues*: CT-2166

Bug Fixes

- Installation and Deployment
  - Fixed RPM package script to run tpm install instead of tpm update when installing the rpm

    *Issues*: CT-2130
- Command-line Tools
  - tpm uninstall would sometimes print `"ERROR >> db1 >> undefined method '+' for nil:NilClass"`

    *Issues*: CT-2104
  - The tpm command now searches more places to locate shell commands that are called, especially useful when `$CONTINUENT_ROOT/share/env.sh` is not sourced.

    *Issues*: CT-2182
- Backup and Restore
  - tprovision will now print an error message and exit if the MySQL datadir does not exit.

    *Issues*: CT-1901
- PostgreSQL Applier
  - Fixes an issue with the `ddl-mysql-postgres.vm` ddlscan template that caused varchar datatype lengths to be parsed incorrectly.

    *Issues*: CT-2019
  - Fixed a `NullPointerException` error occurring at startup when applying to PostgreSQL

    *Issues*: CT-2021
- Core Replicator
  - Improved a query that is run by Tungsten when fetching tables metadata (column names, datatypes, etc). While it is not generally needed, the unoptimized query can run badly (especially) against old mysql versions with a lot of databases / tables. For now, the new optimized query is not used by default, but this could change in some future version.

    This can be enabled by using the following property :

    ```
    property=replicator.datasource.global.connectionSpec.usingOptimizedMetadataQuery=true
    ```

    *Issues*: CT-2077
  - Fixed an issue while processing geometry data with SRID 4326 that would swap longitude and latitude. This applies only to MySQL 8, as prior MySQL versions do not allow specifying the order when applying a WKB (Well-known binary) to MySQL

    *Issues*: CT-2172

# Appendix B. Prerequisites

Before you install Tungsten Cluster, there are a number of setup and prerequisite installation and configuration steps that must have taken place before any installation can continue. Section B.2, "Staging Host Configuration" and Section B.3, "Host Configuration" must be performed on every host within your chosen cluster or replication configuration. Additional steps are required to configure explicit databases, such as Section B.4, "MySQL Database Setup", and will need to be performed on each appropriate host.

## B.1. Requirements

### B.1.1. Operating Systems Support

The following Operating Systems are supported for installation of the Tungsten Replicator and are part of our regular QA testing processes. Other variants of Linux may work at your own risk, but use of them in production should be avoided and any issues that arise may not be supported; if in doubt we recommend that you contact Continuent Support for clarification. Windows/MAC OS is NOT supported for direct installation of Tungsten Replicator, however we do support an "offboard" installation of the replicator on any supported OS to extract/apply to remote databases running on Windows or Mac OS.

Virtual Environments running any of the supported distributions listed are supported, although only recommended for Development/Testing environments.

The list below also includes EOL dates published by the providers and should be taken into consideration when configuring your deployment

Table B.1. Tungsten OS Support

| Distribtion | Published EOL | Notes |
|---|---|---|
| Amazon Linux 2 | 30th June 2024 | |
| Amazon Linux 2023 | | |
| CentOS 7 | 30th June 2024 | |
| Debian GNU/Linux 10 (Buster) | June 2024 | |
| Debian GNU/Linux 11 (Bullseye) | June 2026 | |
| Debian 12 | June 2026 | |
| Oracle Linux 8.4 | July 2029 | |
| RHEL 7 | 30th June 2024 | |
| RHEL 8.4.0 | 31st May 2029 | |
| RHEL 9 | | |
| Rocky Linux 8 | 31st May 2029 | |
| Rocky Linux 9 | 31st May 2032 | |
| SUSE Linux Enterprise Server 15 | 21st June 2028 | |
| Ubuntu 20.04 LTS (Focal Fossa) | April 2025 | |
| Ubuntu 22.04 LTS (Canonical) | April 2027 | |

### B.1.2. Database Support

Unless stated, MySQL refers to the following variants:

- MySQL Community Edition
- MySQL Enterprise Edition
- Percona MySQL

## Version Support Matrix

Table B.2. MySQL/Tungsten Version Support

| Database | MySQL Version | Tungsten Version | Notes |
|---|---|---|---|
| MySQL | 5.7 | All non-EOL Versions | Full Support |
| MySQL | 8.0.0-8.0.34 | 6.1.0-6.1.3 | Supported, but does not support Partitioned Tables or the use of `binlog-transaction-compression=ON` introduced in 8.0.20 |
| MySQL | 8.0.0-8.0.34 | 6.1.4 onwards | Fully Supported. |
| MariaDB | 10.0, 10.1 | All non-EOL Versions | Full Support |
| MariaDB | 10.2, 10.3 | 6.1.13-6.1.20 | Partial Support. See note below. |
| MariaDB | 10.2, 10.3 | 7.x | Full Support |

Known Issue affecting use of MySQL 8.0.21

In MySQL release 8.0.21 the behavior of `CREATE TABLE ... AS SELECT ...` has changed, resulting in the transactions being logged differenly in the binary log. This change in behavior will cause the replicators to fail.

Until a fix is implemented within the replicator, the workaround for this will be to split the action into separate `CREATE TABLE ...` followed by `INSERT INTO ... SELECT FROM...` statements.

If this is not possible, then you will need to manually create the table on all nodes, and then skip the resulting error in the replicator, allowing the subsequent loading of the data to continue.

MariaDB 10.3+ Support

Full support for MariaDB version 10.3 has been certified in v7.0.0 onwards of the Tungsten products.

Version 6.1.13 onwards of Tungsten will also work, however should you choose to deploy these versions, you do so at your own risk. There are a number of issues noted below that are all resolved from v7.0.0 onwards, therefore if you choose to use an earlier release, you should do so with the following limitations acknowledged:

- tungsten_find_orphaned may fail, introducing the risk of data loss (Fixed in v6.1.13 onwards)
- SSL from Tungsten Components TO the MariaDB is not supported.
- Geometry data type is not supported.
- Tungsten backup tools will fail as they rely on xtrabackup, which will not work with newer release of MariaDB.
- tpm might fail to find correct mysql configuration file. (Fixed in 6.1.13 onwards)
- MariaDB specific event types trigger lots of warnings in the replicator log file.

## MySQL "Innovation" Releases

In 2023, Oracle announced a new MySQL version schema that introduced "Innovation" releases. From this point on, patch releases would only contain bug fixes and these would be, for example, the various 8.0.x releases, whereas new features would only be introduced in the "Innovation" releases, such as 8.1, 8.2 etc (Along with Bug Fixes)

"Innovation" releases will be released quartlery, and Oracle aim to make an LTS release every two years which will bundle all of the new features, behavior changes and bug fixes from all the previous "Innovation" releases.

Oracle do not advise the use of the "Innovation" releases in a production enviornment where a known behavior is expected to ensure system stability. We have chosen to follow this advice and as such we do not certify any release of Tungsten against "Innovation" releases for use in Production. We will naturally test against these releases in our QA environment so that we are able to certify and support the LTS release as soon as is practical. Any modifications needed to support an LTS release will not be backported to older Tungsten releases.

For more information on Oracles release policy, please read their blogpost here

# B.1.3. RAM Requirements

RAM requirements are dependent on the workload being used and applied, but the following provide some guidance on the basic RAM requirements:

- Tungsten Replicator requires 2GB of VM space for the Java execution, including the shared libraries, with approximate 1GB of Java VM heapspace. This can be adjusted as required, for example, to handle larger transactions or bigger commit blocks and large packets.

  Performance can be improved within the Tungsten Replicator if there is a 2-3GB available in the OS Page Cache. Replicators work best when pages written to replicator log files remain memory-resident for a period of time, so that there is no file system I/O required to read that data back within the replicator. This is the biggest potential point of contention between replicators and DBMS servers.

## B.1.4. Disk Requirements

Disk space usage is based on the space used by the core application, the staging directory used for installation, and the space used for the THL files:

- The staging directory containing the core installation is approximately 150MB. When performing a staging-directory based installation, this space requirement will be used once. When using a INI-file based deployment, this space will be required on each server. For more information on the different methods, see Section 9.1, "Comparing Staging and `INI` tpm Methods".
- Deployment of a live installation also requires approximately 150MB.
- The THL files required for installation are based on the size of the binary logs generated by MySQL. THL size is typically twice the size of the binary log. This space will be required on each machine in the cluster. The retention times and rotation of THL data can be controlled, see Section D.1.5, "The `thl` Directory" for more information, including how to change the retention time and move files during operation.

A dedicated partition for THL and/or Tungsten Software is recommended to ensure that a full disk does not impact your OS or DBMS. Local disk, SAN, iSCSI and AWS EBS are suitable for storing THL. NFS is NOT recommended.

Because the replicator reads and writes information using buffered I/O in a serial fashion, there is no random-access or seeking.

## B.1.5. Java Requirements

All components of Tungsten are certified with Java using the following versions:

- Oracle JRE 8
- Oracle JRE 11 (From release 6.1.2 only)
- OpenJDK 8
- OpenJDK 11 (From release 6.1.2 only)
- Java 9, 10 and 13 have been tested and validated but certification and support will only cover Long Term releases.

### Important

There are a number of known issues in earlier Java revisions that may cause performance degradation, high CPU, and/or component hangs, specifically when SSL is enabled. It is strongly advised that you ensure your Java version is one of the following MINIMUM releases:

- Oracle JRE 8 Build 261
- OpenJDK 8 Build 222

All versions from 8u265, excluding version 13 onwards, contain a bug that can trigger unusually high CPU and/or system timeouts and hangs within the SSL protocol. To avoid this, you should add the following entry to the `wrapper.conf` file for all relevant components. This will be included by default from version 6.1.15 onwards of all Tungsten products.

`wrapper.conf` can be found in the following path *`{INSTALLDIR}`*/tungsten/tungsten-*`component`*/conf, for example: `/opt/continuent/tungsten/tungsten-manager/conf`

```
wrapper.java.additional.{next available number}=-Djdk.tls.acknowledgeCloseNotify=true
```

For example:

```
wrapper.java.additional.16=-Djdk.tls.acknowledgeCloseNotify=true
```

After editing the file, each component will need restarting

### Important

If your original installation was performed with Java 8 installed, and you wish to upgrade to Java 11, you will need to issue tools/tpm update --replace-release on all nodes from within the software staging path.

> This is to allow the components to detect the newer Java version and adjust to avoid calls to functions that were deprecated/renamed between version 8 and version 11.

## B.1.6. Cloud Deployment Requirements

Cloud deployments require a different set of considerations over and above the general requirements. The following is a guide only, and where specific cloud environment requirements are known, they are explicitly included:

Instance Types/Configuration

| Attribute | Guidance | Amazon Example |
|---|---|---|
| Instance Type | Instance sizes and types are dependent on the workload, but larger instances are recommended for transactional databases. | `m4.xlarge` or better |
| Instance Boot Volume | Use block, not ephemeral storage. | EBS |
| Instance Deployment | Use standard Linux distributions and bases. For ease of deployment and configuration, the use of Ansible, Puppet or other script based solutions could be used. | Amazon Linux AMIs |

Development/QA nodes should always match the expected production environment.

AWS/EC2 Deployments

- Use Virtual Private Cloud (VPC) deployments, as these provide consistent IP address support.
- Multiple EBS-optimized volumes for data, using Provisioned IOPS for the EBS volumes depending on workload:

| Parameter | tpm Option | tpm Value | MySQL `my.cnf` Option | MySQL Value |
|---|---|---|---|---|
| `/` (root) | | | | |
| MySQL Data | `datasource-mysql-data-directory` [401] | `/volumes/mysql/data` | `datadir` | `/volumes/mysql/data` |
| MySQL Binary Logs | `datasource-log-directory` [400] | `/volumes/mysql/binlogs` | `log-bin` | `/volumes/mysql/binlogs/mysql-bin` |
| Transaction History Logs (THL) | `thl-directory` [427] | `/volumes/mysql/thl` | | |

Recommended Replication Formats

- `MIXED` is recommended for MySQL Primary/Replica topologies (e.g., either single clusters or primary/data-recovery setups).
- `ROW` is strongly recommended for Composite Active/Active setups. Without `ROW`, data drift is a possible problem when using `MIXED` or `STATEMENT`. Even with `ROW` there are still cases where drift is possible but the window is far smaller.
- `ROW` is required for heterogeneous replication.

## B.1.7. Docker Support Policy

### B.1.7.1. Overview

Continuent has traditionally had a relaxed policy about Linux platform support for customers using our products.

While it is possible to install and run Continuent Tungsten products (i.e. Clustering/Replicator/etc.) inside Docker containers, there are many reasons why this is not a good idea.

### B.1.7.2. Background

As background, every database node in a Tungsten Cluster runs at least three (3) layers or services:

- MySQL Server (i.e. MySQL Community or Enterprise, MariaDB or Percona Server)
- Tungsten Manager, which handles health-checking, signaling and failover decisions (Java-based)

- Tungsten Replicator, which handles the movement of events from the MySQL Primary server binary logs to the Replica databases nodes (Java-based)

Optionally, a fourth service, the Tungsten Connector (Java-based), may be installed as well, and often is.

### B.1.7.3. Current State

As such, this means that the Docker container would also need to support these 3 or 4 layers and all the resources needed to run them.

This is not what containers were designed to do. In a proper containerized architecture, each container would contain one single layer of the operation, so there would be 3-4 containers per "node". This sort of architecture is best managed by some underlying technology like Swarm, Kubernetes, or Mesos.

More reasons to avoid using Docker containers with Continuent Tungsten solutions:

- Our product is designed to run on a full Linux OS. By design Docker does not have a full init system like SystemD, SysV init, Upstart, etc... This means that if we have a process (Replicator, Manager, Connector, etc...) that process will run as PID 1. If this process dies the container will die. There are some solutions that let a Docker container to have a 'full init' system so the container can start more processes like ssh, replicator, manager, ... all at once. However this is almost a heavyweight VM kind of behavior, and Docker wasn't designed this way.
- Requires a mutable container – to use Tungsten Clustering inside a Docker container, the Docker container must be launched as a mutable Linux instance, which is not the classic, nor proper way to use containers.
- Our services are not designed as "serverless". Serverless containers are totally stateless. Tungsten Cluster and Tungsten Replicator do not support this type of operation.
- Until we make the necessary changes to our software, using Docker as a cluster node results in a minimum 1.2GB docker image.
- Once Tungsten Cluster and Tungsten Replicator have been refactored using a microservices-based architecture, it will be much easier to scale our solution using containers.
- A Docker container would need to allow for updates in order for the Tungsten Cluster and Tungsten Replicator software to be re-configured as needed. Otherwise, a new Docker container would need to be launched every time a config change was required.
- There are known i/o and resource constraints for Docker containers, and therefore must be carefully deployed to avoid those pitfalls.
- We test on CentOS-derived Linux platforms.

### B.1.7.4. Summary

In closing, Continuent's position on container support is as follows:

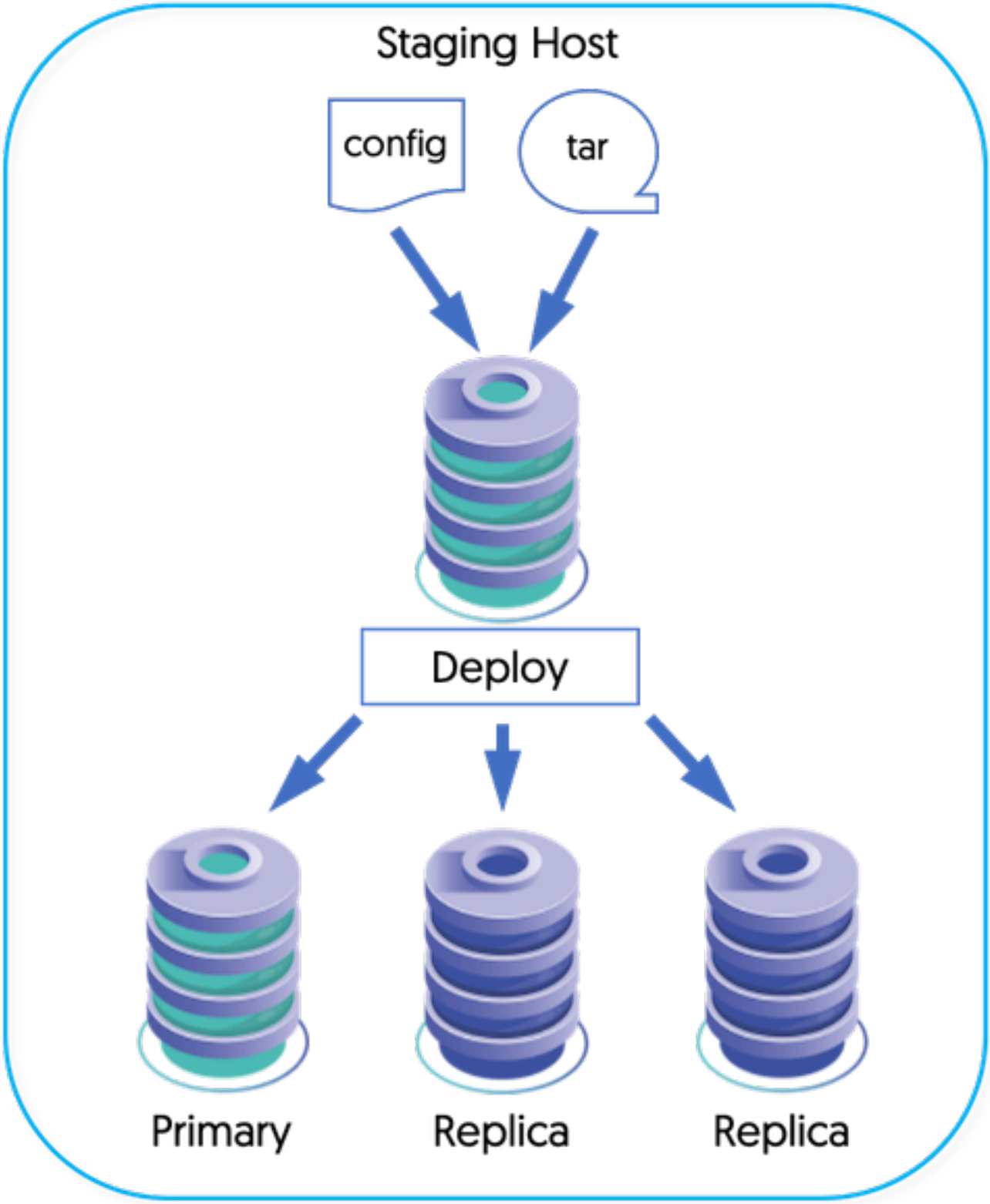
- Unsupported at this time for all products (i.e. Clustering/Replicator/etc.)
- Use at your own risk

## B.2. Staging Host Configuration

The staging host will form the base of your operation for creating your cluster. The primary role of the staging host is to hold the Tungsten Cluster™ software, and to install, transfer, and initiate the Tungsten Cluster™ service on each of the nodes within the cluster. The staging host can be a separate machine, or a machine that will be part of the cluster.

The recommended way to use Tungsten Cluster™ is to configure SSH on each machine within the cluster and allow the tpm tool to connect and perform the necessary installation and setup operations to create your cluster environment, as shown in Figure B.1, "Tungsten Deployment".

Figure B.1. Tungsten Deployment



The staging host will be responsible for pushing and configuring each machine. For this to operate correctly, you should configure SSH on the staging server and each host within the cluster with a common SSH key. This will allow both the staging server, and each host within the cluster to communicate with each other.

You can use an existing login as the base for your staging operations. For the purposes of this guide, we will create a unique user, `tungsten`, from which the staging process will be executed.

1. Create a new Tungsten user that will be used to manage and install Tungsten Cluster™. The recommended choice for MySQL installations is to create a new user, `tungsten`. You will need to create this user on each host in the cluster. You can create the new user using adduser:

   ```
   shell> sudo adduser tungsten
   ```

   You can add the user to the `mysql` group adding the command-line option:

   ```
   shell> sudo usermod -G mysql -a tungsten
   ```

2. Login as the `tungsten` user:

   ```
   shell> su - tungsten
   ```

3. Create an SSH key file, but do not configure a password:

   ```
   tungsten:shell> ssh-keygen -t rsa
   Generating public/private rsa key pair.
   Enter file in which to save the key (/home/tungsten/.ssh/id_rsa):
   Created directory '/home/tungsten/.ssh'.
   Enter passphrase (empty for no passphrase):
   Enter same passphrase again:
   Your identification has been saved in /home/tungsten/.ssh/id_rsa.
   Your public key has been saved in /home/tungsten/.ssh/id_rsa.pub.
   The key fingerprint is:
   e3:fa:e9:7a:9d:d9:3d:81:36:63:85:cb:a6:f8:41:3b tungsten@staging
   The key's randomart image is:
   +--[ RSA 2048]----+
   |                 |
   |                 |
   |             .   |
   |            . .  |
   |        S .. +   |
   |       . o .X .  |
   |        .oEO + . |
   |       .o.=o. o  |
   |      o=+..    . |
   +----------------+
   ```

   This creates both a public and private keyfile; the public keyfile will be shared with the hosts in the cluster to allow hosts to connect to each other.

4. Within the staging server, profiles for the different cluster configurations are stored within a single directory. You can simplify the management of these different services by configuring a specific directory where these configurations will be stored. To set the directory, specify the directory within the `$CONTINUENT_PROFILES` environment variable, adding this variable to your shell startup script (`.bashrc`, for example) within your staging server.

   ```
   shell> mkdir -p /opt/continuent/software/conf
   shell> mkdir -p /opt/continuent/software/replicator.conf
   shell> export CONTINUENT_PROFILES=/opt/continuent/software/conf
   shell> export REPLICATOR_PROFILES=/opt/continuent/software/replicator.conf
   ```

We now have a staging server setup, an SSH keypair for our login information, and are ready to start setting up each host within the cluster.

## B.3. Host Configuration

Each host in your cluster must be configured with the `tungsten` user, have the SSH key added, and then be configured to ensure the system and directories are ready for the Tungsten services to be installed and configured.

There are a number of key steps to the configuration process:

- Creating a user environment for the Tungsten service
- Creating the SSH authorization for the user on each host
- Configuring the directories and install locations
- Installing necessary software and tools

- Configuring sudo access to enable the configured user to perform administration commands

> **Important**
>
> The operations in the following sections must be performed on each host within your cluster. Failure to perform each step may prevent the installation and deployment of Tungsten cluster.

## B.3.1. Creating the User Environment

The `tungsten` user should be created with a home directory that will be used to hold the Tungsten distribution files (not the installation files), and will be used to execute and create the different Tungsten services.

For Tungsten to work correctly, the `tungsten` user must be able to open a larger number of files/sockets for communication between the different components and processes as . You can check this by using ulimit:

```
shell> ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
file size               (blocks, -f) unlimited
max locked memory       (kbytes, -l) unlimited
max memory size         (kbytes, -m) unlimited
open files                      (-n) 256
pipe size            (512 bytes, -p) 1
stack size              (kbytes, -s) 8192
cpu time               (seconds, -t) unlimited
max user processes              (-u) 709
virtual memory          (kbytes, -v) unlimited
```

The system should be configured to allow a minimum of 65535 open files. You should configure both the `tungsten` user and the database user with this limit by editing the `/etc/security/limits.conf` file:

```
tungsten    -    nofile    65535
mysql       -    nofile    65535
```

In addition, the number of running processes supported should be increased to ensure that there are no restrictions on the running processes or threads:

```
tungsten    -    nproc    8096
mysql       -    nproc    8096
```

You must logout and log back in again for the ulimit changes to take effect.

You may also need to set the limit settings on the specific service if your operating system uses the systemctl service management framework. To configure your file limits for the specific service:

1. Copy the MySQL service configuration file template to the configuration directory if it does not already exist:

   ```
   shell> sudo cp /lib/systemd/system/mysql.service /etc/systemd/system/
   ```

   > **Note**
   >
   > Please note that the filename `mysql.service` will vary based on multiple factors. Do check to be sure you are using the correct file. For example, in some cases the filename would be `mysqld.service`

2. Edit the proper file used above, and append to or edit the existing entry to ensure the value of `infinity` for the key `LimitNOFILE`:

   ```
   LimitNOFILE=infinity
   ```

   This configures an unlimited number of open files, you can also specify a number, for example:

   ```
   LimitNOFILE=65535
   ```

3. Reload the systemctl daemon configuration:

   ```
   shell> sudo systemctl daemon-reload
   ```

4. Now restart the MySQL service:

   ```
   shell> service mysql restart
   ```

   > **Warning**
   >
   > On Debian/Ubuntu hosts, limits are not inherited when using su/sudo. This may lead to problems when remotely starting or restarting services. To resolve this issue, uncomment the following line within `/etc/pam.d/su`:

```
session required pam_limits.so
```

Integration with AppArmor

Make sure that Apparmor, if configured, has been enabled to support access to the `/tmp` directory for the MySQL processes. For example, add the following to the MySQL configuration file (usually `/etc/apparmor.d/local/usr.sbin.mysqld`):

```
/tmp/** rwk
```

## B.3.2. Configuring Network and SSH Environment

The hostname, DNS, IP address and accessibility of this information must be consistent. For the cluster to operate successfully, each host must be identifiable and accessible to each other host, either by name or IP address.

Individual hosts within your cluster must be reachable and must conform to the following:

- Do not use the `localhost` or `127.0.0.1` addresses.
- Do not use Zeroconf (`.local`) addresses. These may not resolve properly or fully on some systems.
- The server hostname (as returned by the hostname) must match the names you use when configuring your service.
- The IP address that resolves on the hostname for that host must resolve to the IP address (not `127.0.0.1`). The default configuration for many Linux installations is for the hostname to resolve to the same as `localhost`:

  ```
  127.0.0.1 localhost
  127.0.0.1 host1
  ```

- Each host in the cluster must be able to resolve the address for all the other hosts in the cluster. To prevent errors within the DNS system causing timeouts or bad resolution, all hosts in the cluster, in addition to the witness host, should be added to `/etc/hosts`:

  ```
  127.0.0.1     localhost
  192.168.1.60  host1
  192.168.1.61  host2
  192.168.1.62  host3
  192.168.1.63  host4
  ```

  In addition to explicitly adding hostnames to `/etc/hosts`, the name server switch file, `/etc/nsswitch.conf` should be updated to ensure that hosts are searched first before using DNS services. For example:

  ```
  hosts:          files dns
  ```

  **Important**

  Failure to add explicit hosts and change this resolution order can lead to transient DNS resolving errors triggering timeouts and failsafe switching of hosts within the cluster.

- The IP address of each host within the cluster must resolve to the same IP address on each node. For example, if `host1` resolves to `192.168.0.69` on `host1`, the same IP address must be returned when looking up `host1` on the host `host2`.

To double check this, you should perform the following tests:

1. Confirm the hostname:

   ```
   shell> uname -n
   ```

   **Warning**

   The hostname cannot contain underscores.

2. Confirm the IP address:

   ```
   shell> hostname --ip-address
   ```

3. Confirm that the hostnames of the other hosts in the cluster resolve correctly to a valid IP address. You should confirm on each host that you can identify and connect to each other host in the planned cluster:

   ```
   shell> nslookup host1
   shell> ping host1
   ```

   If the host does not resolve, either ensure that the hosts are added to the DNS service, or explicitly add the information to the `/etc/hosts` file.

### Warning

> If using `/etc/hosts` then you must ensure that the information is correct and consistent on each host, and double check using the above method that the IP address resolves correctly for every host in the cluster.

### B.3.2.1. Network Ports

The following network ports should be open between specific hosts to allow communication between the different components:

| Component | Source | Destination | Port | Purpose |
|---|---|---|---|---|
| All Services | All Nodes | All Nodes | ICMP Ping | Checking availability (Default method) |
| # | # | # | 7 | Checking availability |
| Database Service | Database Host | Database Host | 2112 | THL replication |
| # | # | # | 10000-10001 | Replication connection listener port |

If a system has a firewall enabled, in addition to enabling communication between hosts as in the table above, the localhost must allow port-to-port traffic on the loopback connection without restrictions. For example, using iptables this can be enabled using the following command rule:

```
shell> iptables -A INPUT -i lo -m state --state NEW -j ACCEPT
```

### B.3.2.2. SSH Configuration

For password-less SSH to work between the different hosts in the cluster, you need to copy both the public and private keys between the hosts in the cluster. This will allow the staging server, and each host, to communicate directly with each other using the designated login.

To achieve this, on each host in the cluster:

1. Copy the public (`.ssh/id_rsa.pub`) and private key (`.ssh/id_rsa`) from the staging server to the `~tungsten/.ssh` directory.
2. Add the public key to the `.ssh/authorized_keys` file.

   ```
   shell> cat .ssh/id_rsa.pub >> .ssh/authorized_keys
   ```

3. Ensure that the file permissions on the `.ssh` directory are correct:

   ```
   shell> chmod 700 ~/.ssh
   shell> chmod 600 ~/.ssh/*
   ```

With each host configured, you should try to connecting to each host from the staging server to confirm that the SSH information has been correctly configured. You can do this by connecting to the host using ssh:

```
tungsten:shell> ssh tungsten@host
```

You should have logged into the host at the `tungsten` home directory, and that directory should be writable by the `tungsten` user.

## B.3.3. Directory Locations and Configuration

On each host within the cluster you must pick, and configure, a number of directories to be used by Tungsten Cluster™, as follows:

- `/tmp` Directory

  The `/tmp` directory must be accessible and executable, as it is the location where some software will be extracted and executed during installation and setup. The directory must be writable by the `tungsten` user.

  On some systems, the `/tmp` filesystem is mounted as a separate filesystem and explicitly configured to be non-executable (using the `noexec` filesystem option). Check the output from the mount command.

- Installation Directory

  Tungsten Cluster™ needs to be installed in a specific directory. The recommended solution is to use `/opt/continuent`. This information will be required when you configure the cluster service.

  The directory should be created, and the owner and permissions set for the configured user:

  ```
  shell> sudo mkdir /opt/continuent
  shell> sudo chown -R tungsten: /opt/continuent
  ```

```
shell> sudo chmod 700 /opt/continuent
```

- Home Directory

  The home directory of the `tungsten` user must be writable by that user.

## B.3.4. Configure Software

Tungsten Cluster™ relies on the following software. Each host must use the same version of each tool.

| Software | Versions Supported | Notes |
|---|---|---|
| rsync | - | Check using rsync --help |
| Ruby | 1.8.7, 1.9.3, or 2.0.0 to 2.7.0 [a] | JRuby is not supported |
| Ruby OpenSSL Module | - | Checking using ruby -ropenssl -e 'p "works"' |
| Ruby Gems | - | |
| Ruby `io-console` module | - | Install using gem install io-console [b] |
| Ruby `net-ssh` module | - | Install using gem install net-ssh [c] |
| Ruby `net-scp` module | - | Install using gem install net-scp [d] |
| GNU tar | - | gtar is required for Solaris due to limitations in the native tar command |
| Java Runtime Environment | Java SE 8 (or compatible), Java SE 11 (or compatible) is supported in 6.1.2 and higher | Java 9 and 10 have been tested and validated but certification and support will only cover Long Term releases. See note below for more detail. |
| zip | - | zip is required by tpm diag in 6.1.2 and higher |
| lsb_release | - | lsb_release is required by tpm diag in 6.1.2 and higher. Use sudo yum whatprovides lsb_release to find the appropriate package to install for your operating system. |
| readlink (GNU coreutils) | - | On most platforms, this should be available. readlink, supplied by GNU coreutils, is required by tpm diag |

[a] Ruby 1.9.1 and 1.9.2 are not supported; these releases remove the execute bit during installation.
[b] `io-console` is only needed for SSH activities, and only needed for Ruby v2.0 and greater.
[c] For Ruby 1.8.7 the minimum version of net-ssh is 2.5.2, install using gem install net-ssh -v 2.5.2
[d] For Ruby 1.8.7 the minimum version of net-scp is 1.0.4, install using gem install net-scp -v 1.0.4

These tools must be installed, running, and available to all users on each host.

To check the current version for any installed tool, login as the configured user (e.g. `tungsten`), and execute the command to get the latest version. For example:

- Java

  Run java -version:

```
shell> java -version
openjdk version "1.8.0_102"
OpenJDK Runtime Environment (build 1.8.0_102-b14)
OpenJDK 64-Bit Server VM (build 25.102-b14, mixed mode)
```

> **Important**
>
> See Section B.1.5, “Java Requirements” for more detail on Java requirements and known issues with certain builds.

On certain environments, a separate tool such as alternatives (RedHat/CentOS) or update-alternatives (Debian/Ubuntu) may need to be used to switch Java versions globally or for individual users. For example, within CentOS:

```
shell> alternatives --display
```

> **Important**
>
> It is recommended to switch off all automated software and operating system update procedures. These can automatically install and restart different services which may be identified as failures by Tungsten Replicator. Software and Operating System updates should be handled by following the appropriate Section 7.13, “Performing Database or OS Maintenance” procedures.

It also recommended to install ntp or a similar time synchronization tool so that each host in the cluster has the same physical time.

## B.3.5. sudo Configuration

Tungsten requires that the user you have configured to run the server has sudo credentials so that it can run and install services as `root`.

Within Linux environments you can do this by editing the `/etc/sudoers` file using visudo and adding the following lines:

```
## Allow tungsten to run any command
tungsten ALL=(ALL) NOPASSWD: ALL
```

### Warning

The above syntax is applicable to most Linux environments, however double check if your environment uses different syntax!

sudo can also be configured to handle only specific directories or files. For example, when using xtrabackup, or additional tools in the Tungsten toolkit, such as tprovision, additional commands must be added to the permitted list:

```
tungsten ALL=(ALL) NOPASSWD: /sbin/service, /usr/bin/innobackupex, /bin/rm, »
    /bin/mv, /bin/chown, /bin/chmod, /usr/bin/scp, /bin/tar, /usr/bin/which, »
    /etc/init.d/mysql, /usr/bin/test, /usr/bin/systemctl, »
    /opt/continuent/tungsten/tungsten-replicator/scripts/xtrabackup.sh, »
    /opt/continuent/tungsten/tools/tpm, /usr/bin/innobackupex-1.5.1, »
    /bin/cat, /bin/find, /usr/bin/whoami, /bin/sh, /bin/rmdir, /bin/mkdir, »
    /usr/bin/mysql_install_db, /usr/bin/mysqld, /usr/bin/xtrabackup
```

### Note

On some versions of sudo, use of sudo is deliberately disabled for ssh sessions. To enable support via ssh, comment out the requirement for `requiretty`:

```
#Defaults    requiretty
```

## B.3.6. SELinux Configuration

To determine the surrent state of SELinux enforcement, use the getenforce command. For example:

```
shell> getenforce
Disabled
```

To disable SELinux, use the setenforce command. For example:

```
shell> setenforce 0
```

Should your company policy enforce the use of SELinux, then you will need to configure various SELinux contexts to allow Tungsten to operate.

When SELinux is enabled, systemctl may refuse to start mysqld if the listener port or location on disk have been changed. The solution is to inform SELinux about any changed or additional resources.

Tungsten best practice is to change the default MySQL port from `3306` to `13306` so that requesting clients do not accidentally connect directly to the database without being routed by the Connector.

If using a non-standard port for MySQL and SELinux is enabled, you must also change the port context, for example:

```
shell > semanage port -a -t mysqld_port_t -p tcp 13306
```

Ensure the file contexts are set correctly for SELinux. For example, to allow MySQL data to be stored in a non-standard location (i.e. `/data`):

```
shell > semanage fcontext -a -t etc_runtime_t /data
shell > restorecon -Rv /data/

shell > semanage fcontext -a -t mysqld_db_t "/data(/.*)?"
shell > restorecon -Rv /data/*
```

# B.4. MySQL Database Setup

For replication between MySQL hosts, you must configure each MySQL database server to support the required user names and core MySQL configuration.

### Important

For MySQL extraction, Tungsten Cluster must have write access to the database so that status and progress information can be recorded correctly.

### Note

Native MySQL replication should not be running when you install Tungsten Cluster™. The replication service will be completely handled by Tungsten Cluster™, and the normal replication, management and monitoring techniques will not provide you with the information you need.

## B.4.1. MySQL Version Support

For a full list of MySQL Versions supported, see Table B.2, "MySQL/Tungsten Version Support"

## B.4.2. MySQL Configuration

Each MySQL Server should be configured identically within the system. Although binary logging must be enabled on each host, replication should not be configured, since Tungsten Replicator will be handling that process.

The configured `tungsten` user must be able to read the MySQL configuration file (for installation) and the binary logs. Either the `tungsten` user should be a member of the appropriate group (i.e. `mysql`), or the permissions altered accordingly.

### Important

Parsing of mysqld_multi configuration files is not currently supported. To use a mysqld_multi installation, copy the relevant portion of the configuration file to a separate file to be used during installation.

To setup your MySQL servers, you need to do the following:

- Configure your `my.cnf` settings. The following changes should be made to the `[mysqld]` section of your `my.cnf` file:
  - By default, MySQL is configured only to listen on the localhost address (127.0.0.1). The `bind-address` parameter should be checked to ensure that it is either set to a valid value, or commented to allow listening on all available network interfaces:

    ```
    #bind-address = 127.0.0.1
    ```

  - Specify the server id

    Each server must have a unique server id:

    ```
    server-id = 1
    ```

    The best practice is for all servers to have a unique ID across all clusters. For example, use a numbering scheme like 0101, 0102, 0201, 0201, where the leading two digits are the cluster number and the last two digits are the node number, which allows for 99 participating clusters with 99 nodes each.

  - Ensure that the maximum number of open files matches the configuration of the database user. This was configured earlier at 65535 files.

    ```
    open_files_limit = 65535
    ```

  - Enable binary logs

    Tungsten Replicator operates by reading the binary logs on each machine, so logging must be enabled:

    ```
    log-bin = mysql-bin
    ```

  - Set the `sync_binlog` parameter to 1 (one).

    #### Note

    In MySQL 5.7, the default value is 1.

    The MySQL `sync_binlog` parameter sets the frequency at which the binary log is flushed to disk. A value of zero indicates that the binary log should not be synchronized to disk, which implies that only standard operating system flushing of writes will occur. A value greater than one configures the binary log to be flushed only after `sync_binlog` events have been written. This can introduce a delay into writing information to the binary log, and therefore replication, but also opens the system to potential data loss if the binary log has not been flushed when a fatal system error occurs.

    Setting a value of value 1 (one) will synchronize the binary log on disk after each event has been written.

    ```
    sync_binlog = 1
    ```

  - Increase MySQL protocol packet sizes

The replicator can apply statements up to the maximum size of a single transaction, so the maximum allowed protocol packet size must be increase to support this:

```
max_allowed_packet = 52m
```

- Configure InnoDB as the default storage engine

  Tungsten Cluster needs to use a transaction safe storage engine to ensure the validity of the database. The InnoDB storage engine also provides automatic recovery in the event of a failure. Using MyISAM can lead to table corruption, and in the event of a switchover or failure, and inconsistent state of the database, making it difficult to recover or restart replication effectively.

  InnoDB should therefore be the default storage engine for all tables, and any existing tables should be converted to InnoDB before deploying Tungsten Cluster.

```
default-storage-engine = InnoDB
```

- Configure InnoDB Settings

  Tungsten Replicator creates tables and must use InnoDB tables to store the status information for replication configuration and application:

  The MySQL option `innodb_flush_log_at_trx_commit` configures how InnoDB writes and confirms writes to disk during a transaction. The available values are:

  - A value of 0 (zero) provides the best performance, but it does so at the potential risk of losing information in the event of a system or hardware failure. For use with Tungsten Cluster™ the value should never be set to 0, otherwise the cluster health may be affected during a failure or failover scenario.

  - A value of 1 (one) provides the best transaction stability by ensuring that all writes to disk are flushed and committed before the transaction is returned as complete. Using this setting implies an increased disk load and so may impact the overall performance.

    When using Tungsten Cluster in an Composite Active/Active, fan-in or data critical cluster, the value of `innodb_flush_log_at_trx_commit` should be set to 1. This not only ensures that the transactional data being stored in the cluster are safely written to disk, this setting also ensures that the metadata written by Tungsten Cluster™ describing the cluster and replication status is also written to disk and therefore available in the event of a failover or recovery situation.

  - A value of 2 (two) ensures that transactions are committed to disk, but data loss may occur if the disk data is not flushed from any OS or hardware-based buffering before a hardware failure, but the disk overhead is much lower and provides higher performance.

    This setting must be used as a minimum for *all* Tungsten Cluster™ installations, and should be the setting for all configurations that do not require `innodb_flush_log_at_trx_commit` set to `1`.

  At a minimum `innodb_flush_log_at_trx_commit` should be set to `2`; a warning will be generated if this value is set to zero:

```
innodb_flush_log_at_trx_commit = 2
```

  MySQL configuration settings can be modified on a running cluster, providing you switch your host to maintenance mode before reconfiguring and restarting MySQL Server. See Section 7.13, “Performing Database or OS Maintenance”.

Optional configuration changes that can be made to your MySQL configuration:

- InnoDB Flush Method

```
innodb_flush_method=O_DIRECT
```

  The InnoDB flush method can effect the performance of writes within MySQL and the system as a whole.

  O_DIRECT is generally recommended as it eliminates double-buffering of InnoDB writes through the OS page cache. Otherwise, MySQL will be contending with Tungsten and other processes for pages there — MySQL is quite active and has a lot of hot pages for indexes and the like this can result lower i/o throughput for other processes.

  Tungsten particularly depends on the page cache being stable when using parallel apply. There is one thread that scans forward over the THL pages to coordinate the channels and keep them from getting too far ahead. We then depend on those pages staying in cache for a while so that all the channels can read them — as you are aware parallel apply works like a bunch of parallel table scans that are traveling like a school of sardines over the same part of the THL. If pages get kicked out again before all the channels see them, parallel replication will start to serialize as it has to wait for the OS to read them back in again. If they stay in memory on the other hand, the reads on the THL are in-memory, and fast. For more information on parallel replication, see Section 5.5, “Deploying Parallel Replication”.

- Increase InnoDB log file size

The default InnoDB Redo Log file size is 48MB. This should be increased to a larger file size for performance and other reasons. Values of 512MB are common.

To change the file size, read the corresponding information in the MySQL manual for configuring the file size information. Please see both "MySQL Redo Log" and "Optimizing MySQL InnoDB Redo Logging".

- Binary Logging Format

  Tungsten Replicator works with both statement and row-based logging, and therefore also mixed-based logging. The chosen format is entirely up to the systems and preferences, and there are no differences or changes required for Tungsten Replicator to operate. For native MySQL to MySQL Primary/Replica replication, either format will work fine.

  Depending on the exact use case and deployment, different binary log formats imply different requirements and settings. Certain deployment types and environments require different settings:

  - For Composite Active/Active deployments, use row-based logging. This will help to avoid data drift where statements make fractional changes to the data in place of explicit updates.
  - Use row-based logging for heterogeneous deployments. All deployments to Oracle, MongoDB, Vertica and others rely on row-based logging.
  - Use mixed replication if warnings are raised within the MySQL log indicating that statement only is transferring possibly dangerous statements.
  - Use statement or mixed replication for transactions that update many rows; this reduces the size of the binary log and improves the performance when the transaction are applied on the Replica.
  - Use row replication for transactions that have temporary tables. Temporary tables are replicated if statement or mixed based logging is in effect, and use of temporary tables can stop replication as the table is unavailable between transactions. Using row-based logging also prevents these tables entering the binary log, which means they do not clog and delay replication.

  The configuration of the MySQL server can be permanently changed to use an explicit replication by modifying the configuration in the configuration file:

  ```
  binlog-format = row
  ```

  > **Note**
  >
  > In MySQL 5.7, the default format is `ROW`.

  For temporary changes during execution of explicit statements, the binlog format can be changed by executing the following statement:

  ```
  mysql> SET binlog-format = ROW;
  ```

- **`innodb_stats_on_metadata=0`**

  Although optional, we would highly recommend setting this property as it has been shown to improve performance by preventing statistics updates every time the information_schema is queried.

  You must restart MySQL after any changes have been made.

- Ensure the `tungsten` user can access the MySQL binary logs by either opening up the directory permissions, or adding the `tungsten` user to the group owner for the directory.

## B.4.3. MySQL Configuration for Active/Active Deployments

If you are inserting to the same table at the same time at two or more different sites, and using bi-directional or active/active replication, then special care must be taken to avoid primary key conflicts. Either the auto-increment keys on each need to be offset so they do not conflict, or the application needs to be be able to generate unique keys taking multiple sites into account.

> **Important**
>
> The following configuration is *required* if your application is relying upon the MySQL-native auto-increment primary key feature:

Use the `auto-increment-increment` and `auto-increment-offset` variables to affect the way that MySQL generates the next value in an auto-increment field.

For example, edit `my.cnf` on all servers:

```
# for all servers at site 1

auto-increment-increment = 10
auto-increment-offset = 1
```

```
# for all servers at site 2

auto-increment-increment = 10
auto-increment-offset = 2
```

```
# for all servers at site 3

auto-increment-increment = 10
auto-increment-offset = 3
```

**Important**

Restart MySQL on all servers.

## B.4.4. MySQL Configuration for Heterogeneous Deployments

The following are *required* for replication to heterogeneous targets to ensure that MySQL has been configured and generating row change information correctly:

- MySQL must be using Row-based replication for information to be replicated to heterogenous targets. For the best results, you should change the global binary log format, ideally in the configuration file (`my.cnf`):

  ```
  binlog-format = row
  ```

  Alternatively, the global binlog format can be changed by executing the following statement:

  ```
  mysql> SET GLOBAL binlog-format = ROW;
  ```

  For MySQL 5.6.2 and later, you must enable full row log images:

  ```
  binlog-row-image = full
  ```

  This information will be forgotten when the MySQL server is restarted; placing the configuration in the `my.cnf` file will ensure this option is permanently enabled.

- Table format should be updated to UTF8 by updating the MySQL configuration (`my.cnf`):

  ```
  character-set-server=utf8
  collation-server=utf8_general_ci
  ```

  Tables must also be configured as UTF8 tables, and existing tables should be updated to UTF8 support before they are replicated to prevent character set corruption issues.

- To prevent timezone configuration storing zone adjusted values and exporting this information to the binary log and PostgreSQL, fix the timezone configuration to use UTC within the configuration file (`my.cnf`):

  ```
  default-time-zone='+00:00'
  ```

## B.4.5. MySQL User Configuration

- Tungsten User Login

  It is possible to use users with a lower-privilege level and without as many rights. For more information, see Section B.4.6, “MySQL Unprivileged Users”.

  The `tungsten` user connects to the MySQL database and applies the data from the replication stream from other datasources in the dataservice. The user must therefore be able execute any SQL statement on the server, including grants for other users. The user *must* have the following privileges in addition to privileges for creating, updating and deleting DDL and data within the database:

  - `SUPER` privilege is required so that the user can perform all administrative operations including setting global variables.
  - `GRANT OPTION` privilege is required so that users and grants can be updated.

  To create a user with suitable privileges:

  ```
  mysql> CREATE USER tungsten@'%' IDENTIFIED BY 'password';
  mysql> GRANT ALL ON *.* TO tungsten@'%'  WITH GRANT OPTION;
  ```

  The connection will be made from the host to the local MySQL server. You may also need to create an explicit entry for this connection. For example, on the host `host1`, create the user with an explicit host reference:

```
mysql> CREATE USER tungsten@'host1' IDENTIFIED BY 'password';
mysql> GRANT ALL ON *.* TO tungsten@'host1'  WITH GRANT OPTION;
```

The above commands enable logins from any host using the user name/password combination. If you want to limit the configuration to only include the hosts within your cluster you must create and grant individual user/host combinations:

```
mysql> CREATE USER tungsten@'client1' IDENTIFIED BY 'password';
mysql> GRANT ALL ON *.* TO tungsten@'client1'  WITH GRANT OPTION;
```

**Note**

> If you later change the cluster configuration and add more hosts, you will need to update this configuration with each new host in the cluster.

- If you configure the connector to run in Proxy mode, and you issue the `SHOW SLAVE STATUS` command, then any user executing this statement will require the select privilege on the tracking schema table `trep_commit_seqno`. The following DDL can be used as an example:

```
GRANT SELECT ON tungsten_<servicename>.trep_commit_seqno TO '<user>'@'<host>';
```

This will need to be executed after installation, following the initial creation of the tracking schema and tables.

## B.4.6. MySQL Unprivileged Users

By default, the `tungsten` user needs to be given `SUPER` privileges within MySQL so that the user can apply, create and access all the tables and data within the MySQL database. In some situations, this level of access is not available within the MySQL environment, for example, when using a server that is heavily secured, or Amazon's RDS service.

For this situation, the Tungsten Cluster can be configured to use an 'unprivileged' user configuration. This configuration does not require the `SUPER` privilege, but instead needs explicit privileges on the schema created by Tungsten Cluster, and on the schemas that it will update when applying events.

The capability can be enabled by using the following two options and behaviors:

- `--privileged-master=false` [417]

  When privileged_master is disabled:

  - A Primary replicator will not attempt to suppress binlog writes during operations.
  - A Primary replicator Will not issue a `FLUSH LOGS` command when the replicator starts.
  - The current replicator position is not updated within the `trep_commit_seqno` table.

  The `tungsten` user that connects to the database must be configured to work with the MySQL service using the following grants:

```
mysql> GRANT ALL ON tungsten_alpha.* to tungsten@'%' IDENTIFIED BY 'secret';
mysql> GRANT SELECT ON *.* TO tungsten@'%' IDENTIFIED BY 'secret';
mysql> GRANT REPLICATION SLAVE ON *.* TO tungsten@'%' IDENTIFIED BY 'secret';
mysql> REVOKE SUPER ON *.* FROM tungsten@'%';
```

- `--privileged-slave=false` [417]

  When privileged_slave is disabled:

  - The current replicator position is not updated within the `trep_commit_seqno` table.

```
mysql> GRANT ALL ON tungsten_batch.* to tungsten@'%' IDENTIFIED BY 'secret';
mysql> GRANT SELECT,INSERT,UPDATE ON *.* TO tungsten@'%' IDENTIFIED BY 'secret';
mysql> GRANT REPLICATION SLAVE ON *.* TO tungsten@'%' IDENTIFIED BY 'secret';
mysql> REVOKE SUPER ON *.* FROM tungsten@'%';
```

  Optionally, `INSERT` and `UPDATE` privileges can be explicitly added to the user permissions for the tables/databases that will be updated during replication.

# B.5. Prerequisite Checklist

To simplify the process of preparing your hosts, the checklist below is designed to provide a quick summary of the main prerequisites required.

A PDF version of this checklist can also be downloaded here

Host Specific

| Pre-Req | Complete? |
|---|---|
| Create OS User – Typically called `tungsten` | |
| Set `ulimit` for OS User | |
| Configure sudoers | |
| Disable SELinux | |
| Compile `/etc/hosts` | |
| Setup SSH between hosts | |
| Create directory for installation (Typically, `/opt/continuent`) | |
| Create directory for software package if using tar bundle (Typically, `/opt/continuent/software`) | |
| Create directory for configuration file if INI Install (`/etc/tungsten`) | |
| Check ownership of new directories set to new OS user | |
| Install Ruby | |
| Install Ruby gems : `net-ssh` | |
| Install Ruby gems : `net-scp` | |
| Install Ruby gems : `io-console` | |
| Install Java 8 | |
| Install rsync | |

Network Specific

| Pre-Req | Complete? |
|---|---|
| Ensure Network ports open | |

Database Specific (All Topologies)

| Pre-Req | Complete? |
|---|---|
| Ensure `server-id` unique amongst all nodes | |
| Increase Open Files limits | |
| Ensure `bin-logging` enabled for cluster nodes, or source replicator nodes | |
| Review `sync_binlog` parameter | |
| Increase, if required, `max_allowed_packet` | |
| Review InnoDB settings | |
| Set `binlog_format` to ROW (Essential for Active/Active or heterogeneous deployments) | |
| Ensure `auto_increment` offsets adjusted for Active/Active deployments | |
| Create DB user with FULL privileges and `GRANT OPTION` – typically called `tungsten` (Used by managers and replicators) | |

# Appendix C. Troubleshooting

The following sections contain both general and specific help for identifying, troubleshooting and resolving problems. Key sections include:

- General notes on contacting and working with support and supplying information, see Section C.1, “Contacting Support”.
- Error/Cause/Solution guidance on specific issues and error messages, and how the reason can be identified and resolved, see Section C.3, “Error/Cause/Solution”.
- Additional troubleshooting for general systems and operational issues.

## C.1. Contacting Support

The support portal may be accessed at https://continuent.zendesk.com.

Continuent offers paid support contracts for Continuent Tungsten and Tungsten Replicator. If you are interested in purchasing support, contact our sales team at sales@continuent.com.

### C.1.1. Support Request Procedure

Please use the following procedure when requesting support so we can provide prompt service. If we are unable to understand the issue due to lack of required information, it will prevent us from providing a timely response.

1. Please provide a clear description of the problem
2. Which environment is having the issue? (Prod, QA, Dev, etc.)
3. What is the impact upon the affected environment?
4. Identify the problem host or hosts and the role (Primary, Replica, etc)
5. Provide the steps you took to see the problem in your environment
6. Upload the resulting zip file from tpm diag, potentially run more than once on different hosts as needed. Alternatively, use the tungsten_send_diag command.
7. Provide steps already taken and commands already run to resolve the issue
8. Have you searched your previous support cases? https://continuent.zendesk.com.
9. Have you checked the Continuent documentation? https://docs.continuent.com
10. Have you checked our general knowledge base? For our Error/Cause/Solution guidance on specific issues and error messages, and how the reason can be identified and resolved, see Section C.3, “Error/Cause/Solution”.

### C.1.2. Creating a Support Account

You can create a support account by logging into the support portal at https://continuent.zendesk.com. Please use your work email address so that we can recognize it and provide prompt service. If we are unable to recognize your company name it may delay our ability to provide a response.

Be sure to allow email from `helpdesk@continuent.com` and `notifications-helpdesk@continuent.com`. These addresses will be used for sending messages from Zendesk.

### C.1.3. Open a Support Ticket

Login to the support portal and click on 'Submit a Request' at the top of the screen. You can access this page directly at https://continuent.zendesk.com/requests/new.

### C.1.4. Open a Support Ticket via Email

Send an email to helpdesk@continuent.com from the email address that you used to create your support account. You can include a description and attachments to help us diagnose the problem.

### C.1.5. Getting Updates for all Company Support Tickets

If multiple people in your organization have created support tickets, it is possible to get updates on any support tickets they open. You should see your organization name along the top of the support portal. It will be listed after the Check Your Existing Requests tab.

To see all updates for your organization, click on the organization name and then click the Subscribe link.

If you do not see your organization name listed in the headers, open a support ticket asking us to create the organization and list the people that should be included.

### C.1.6. Support Severity Level Definitions

Summary of the support severity levels with initial response targets:

- Urgent: initial response within an hour

  Represents a reproducible emergency condition (i.e. a condition that involves either data loss, data corruption, or lack of data availability) that makes the use or continued use of any one or more functions impossible. The condition requires an immediate solution. Continuent guarantees a maximum one (1) hour initial response time. Continuent will continue to work with Customer until Customer's database is back in production. The full resolution and the full root cause analysis will be provided when available.

- High: initial response within four (4) hours

  Represents a reproducible, non-emergency condition (i.e. a condition that does not involve either data loss, data corruption or lack of database availability) that makes the use or continued use of any one or more functions difficult, and cannot be circumvented or avoided on a temporary basis by Customer. Continuent guarantees a maximum four (4) hours initial response time.

- Normal: initial response within one (1) business day

  Represents a reproducible, limited problem condition that may be circumvented or avoided on a temporary basis by Customer. Continuent guarantees a maximum one (1) business day initial response time.

- Low: no guaranteed initial response interval

  Represents minor problem conditions or documentation errors that are easily avoided or circumvented by Customer. Additional request for new feature suggestions, which are defined as new functionality in existing product, are also classified as low severity level. Continuent does not guarantee any particular initial response time, or a commitment to fix in any particular time frame unless Customer engages Continuent for professional services work to create a fix or a new feature.

## C.2. Support Tools

### C.2.1. Generating Diagnostic Information

To aid in the diagnosis of issues, a copy of the logs and diagnostic information will help the support team to identify and trace the problem. There are two methods of providing this information:

- Using tpm diag

  The tpm diag command will collect the logs and configuration information from the active installation and generate a Zip file with the diagnostic information for all hosts within it. The command should be executed from the staging directory. Use tpm query staging to determine this directory:

  ```
  shell> tpm query staging
  tungsten@host1:/home/tungsten/tungsten-replicator-7.1.2-81
  shell> cd /home/tungsten/tungsten-replicator-7.1.2-81
  shell> ./tools/tpm diag
  ```

  The process will create a file called `tungsten-diag-2014-03-20-10-21-29.zip`, with the corresponding date and time information replaced. This file should be included in the reported support issue as an attachment.

  For a staging directory installation, tpm diag will collect together all of the information from each of the configured hosts in the cluster. For an INI file based installation, tpm diag will connect to all configured hosts if ssh is available. If a warning that ssh is not available is generated, tpm diag must be run individually on each host in the cluster.

- Manually Collecting Logs

  If tpm diag cannot be used, or fails to return all the information, the information can be collected manually:

  1. Run tpm reverse on all the hosts in the cluster:

     ```
     shell> tpm reverse
     ```

  2. Collect the logs from each host. Logs are available within the `service_logs` directory. This contains symbolic links to the actual log files. The original files can be included within a tar archive by using the `-h` option. For example:

```
shell> cd /opt/continuent
shell> tar zcfh host1-logs.tar.gz ./service_logs
```

The tpm reverse and log archives can then be submitted as attachments with the support query.

## C.2.2. Generating Advanced Diagnostic Information

To aid in the diagnosis of difficult issues, below are tools and procedures to assist in the data collection.

> **Warning**
>
> ONLY excute the below commands and procedures when requested by Continuent support staff.

### Manager Memory Usage Script

We have provided a script to easily tell us how much memory a given manager is consuming.

Place the script on all of your manager hosts (i.e. into the tungsten OS user home directory).

> **Note**
>
> The script assumes that 'cctrl' is in the path. If not, then change the script to provide a full path for cctrl.

```
shell> su - tungsten
shell> vi tungsten_manager_memory
#!/bin/bash
memval=`echo gc | cctrl | grep used | tail -1 | awk -F: '{print $2}' | tr -d ' |'`
megabytes=`expr $memval / 1000000`
timestamp=`date +"%F %T" | tr '-' '/'`
echo "$timestamp | `hostname` | $megabytes MB"

shell> chmod 750 tungsten_manager_memory
shell> ./tungsten_manager_memory
```

This script is ideally run from cron and the output redirected to time-stamped log files for later correlation with manager issues.

### Manager Thread Dump Procedure

This procedure creates a Manager memory thread dump for detailed analysis.

Run this command on manager hosts when requested by Continuent support.

This will append the detailed thread dump information to the log file named `tmsvc.log` in the `/opt/continuent/tungsten/tungsten-manager/log` directory.

```
shell> su - tungsten
shell> manager dump
shell> tungsten_send_diag -f /opt/continuent/tungsten/tungsten-manager/log/tmsvc.log -c {case_number}
```

### Manager Heap Dump Procedure

This procedure creates a Manager memory heap dump for detailed analysis.

Run this command on manager hosts when requested by Continuent support.

This will create a file named `{hostname}.hprof` in the directory where you run it.

```
shell> su - tungsten
shell> jmap -dump:format=b,file=`hostname`.hprof `ps aux | grep JANINO | grep -v grep | awk '{print $2}'`
shell> tungsten_send_diag -f `hostname`.hprof -c {case_number}
```

### Configuring Connector Debug Logging

This procedure allows the Connector to be configured for debug logging.

Perform this procedure on Connector hosts when requested by Continuent support.

> **Warning**
>
> Enabling Connector debug logging will decrease performance dramatically. Disk writes will increase as will disk space consumption. Do not use in production environments unless instructed to do so by Continuent support. In

> any case, run in this mode for as short a period of time as possible - just long enough to gather the needed debug information. After that is done, disable debug logging.

To enable debug logging, edit the Connector configuration file `tungsten-connector/conf/log4j.properties` and turn Connector logging from INFO to DEBUG (or to TRACE for verbose logging):

```
shell> su - tungsten
shell> vi /opt/continuent/tungsten/tungsten-connector/conf/log4j.properties
# Enable debug for the connector only
logger.Connector.name=com.continuent.tungsten.connector
# WAS: logger.Connector.level=INFO
logger.Connector.level=DEBUG
logger.Connector.additivity=false
shell> connector reconfigure
```

To disable debug logging, edit the Connector configuration file `tungsten-connector/conf/log4j.properties` and revert the change from `DEBUG` to `INFO`.

## C.2.3. Using tungsten_upgrade_manager

tungsten_upgrade_manager is used to correct a cctrl display bug in the Manager that causes the `useSSL` value shown via `cctrl> ls -l` to be false when it should be true after an upgrade from v6 to v7.

### Warning

> Only use the tungsten_upgrade_manager command when instructed to do so by Continuent Support!

# C.3. Error/Cause/Solution

## C.3.1. MySQLExtractException: unknown data type 0

*Last Updated: 2014-04-15*

*Condition or Error*

Replication fails to extract the data from the MySQL binary log, and the replicator will not go online again.

*Causes*

- The format of `DECIMAL` types between MySQL 4.x and MySQL 5.0 changed, however, the datatype was not automatically modified during an upgrade process. This means that tables that were created in MySQL 4.x and now exist within MySQL 5.0 using the `DECIMAL` generate an incompatible entry within the MySQL binary log. The upgrade and mysql_upgrade commands do not update the tables correctly. More detailed information on the change and issue can be located in Bug #57166.

*Rectifications*

- The table definition must be manually upgraded to force the change of the columns using the older `DECIMAL` type. The recommended correction is to explicitly upgrade the `DECIMAL` columns. For example:

  ```
  mysql> ALTER TABLE faulty MODIFY COLUMN faulty_column DECIMAL;
  ```

  This should be performed on the Primary within your topology. To correct the error, you must use tpm reset-thl to regenerate the THL.

## C.3.2. Services requires a reset

*Last Updated: 2016-05-18*

*Condition or Error*

The replicator service needs to be reset, for example if your MySQL service has been reconfigured, or when resetting a data warehouse or batch loading service after a significant change to the configuration.

*Causes*

- If the replicator stops replicating effectively, or the configuration and/or schema of a source or target in a datawarehouse loading solution has changed significantly. This will reset the service, starting extraction from the current point, and the target/Replica from the new Primary position. It will also reset all the positions for reading and writing.

*Rectifications*

- To reset a service entirely, without having to perform a re-installation, you should follow these steps. This will reset both the THL, source database binary log reading position and the target THL and starting point.

  1. Take the Replica offline:

     ```
     Replica-shell> trepctl offline
     ```

  2. Take the Primary offline:

     ```
     Replica-shell> trepctl offline
     ```

  3. Use trepctl to reset the service on the Primary and Replica. You must use the service name explicitly on the command-line:

     ```
     Primary-shell> trepctl -service alpha reset -y
     Replica-shell> trepctl -service alpha reset -y
     ```

  4. Put the Replica online:

     ```
     Replica-shell> trepctl offline
     ```

  5. Put the Primary online:

     ```
     Replica-shell> trepctl offline
     ```

## C.3.3. OptimizeUpdatesFilter cannot filter, because column and key count is different. Make sure that it is defined before filters which remove keys (eg. PrimaryKeyFilter)

*Last Updated: 2014-07-28*

*Condition or Error*

When using the `optimizeupdates` filter, replication stops with the error message in the output from trepctl status or when examining the log file.

*Causes*

- The `optimizeupdates` filter works by removing indexed columns from updates that are unnecessary when a primary key exists to locate the record. If the key information has already been removed (for example, by the `pkey` filter, then the columns cannot be effectively compared and optimized.

*Rectifications*

- If the `pkey` filter is required, change the order of the filters within the specified stage within the replicator so that the `optimizeupdates` filter is called *before* the `pkey` filter.

*More Information*

Section 11.4.31, “PrimaryKey Filter”

## C.3.4. Unable to update the configuration of an installed directory

*Last Updated: 2013-08-07*

*Condition or Error*

Running an update or configuration with tpm returns the error 'Unable to update the configuration of an installed directory'

*Causes*

- Updates to the configuration of a running cluster must be performed from the staging directory where Tungsten Cluster was originally installed.

*Rectifications*

- Change to the staging directory and perform the necessary commands with tpm. To determine the staging directory, use:

  ```
  shell> tpm query staging
  ```

  Then change to the staging directory and perform the updates:

  ```
  shell> ./tools/tpm configure ....
  ```

*More Information*

Chapter 2, *Deployment Overview*

## C.3.5. Too many open processes or files

*Last Updated: 2013-10-09*

*Condition or Error*

The operating system or environment reports that the `tungsten` or designated Tungsten Cluster user has too many open files, processes, or both.

*Causes*

- User limits for processes or files have either been exhausted, or recommended limits for user configuration have not been set.

*Rectifications*

- Check the output of ulimit and check the configure file and process limits:

```
shell> ulimit -a
core file size (blocks, -c) 0
data seg size (kbytes, -d) unlimited
file size (blocks, -f) unlimited
max locked memory (kbytes, -l) unlimited
max memory size (kbytes, -m) unlimited
open files (-n) 256
pipe size (512 bytes, -p) 1
stack size (kbytes, -s) 8192
cpu time (seconds, -t) unlimited
max user processes (-u) 709
virtual memory (kbytes, -v) unlimited
```

  If the figures reported are less than the recommended settings, see Section B.3.1, “Creating the User Environment” for guidance on how these values should be changed.

*More Information*

Section B.3.1, “Creating the User Environment”

## C.3.6. There were issues configuring the sandbox MySQL server

*Last Updated: 2016-04-20*

*Condition or Error*

- The command tungsten_provision_thl fails when using Percona Server.
- When running the command tungsten_provision_thl, you see the error:

```
There were issues configure the sandbox MySQL server
```

- MySQL Sandbox fails when using Percona Server.
- In the `$CONTINUENT_ROOT/service_logs/provision_thl.log` file, you see entries similar to:

```
mysqld: error while loading shared libraries: libssl.so.6: cannot open shared object file: No such file or directory
```

- In the `$CONTINUENT_ROOT/provision_thl.log` file, you see entries similar to:

```
mysql_install_db Error in my_thread_global_end(): 1 threads didn't exit
```

*Causes*

- This issue occurs because of a problem in Percona Server tarball distributions.

  There are two issues with Percona Server tarball distributions, which depends on the version you have downloaded.

  Look in the log file $CONTINUENT_ROOT/service_logs/provision_thl.log for:

  - `mysqld: error while loading shared libraries: libssl.so.6`
  - `mysql_install_db Error in my_thread_global_end()`

*Rectifications*

- To resolve this issue in Centos, install openssl by running the command:

```
shell> sudo yum install openssl098e
```

  Alternatively, use Oracle MySQL or MariaDB which do not experience these issues.

  **Note**

  VMware does not endorse or recommend any particular third party utility.

*More Information*

Section 8.30, "The tungsten_provision_thl Command"

## C.3.7. Unexpected failure while extracting event

*Last Updated: 2020-10-13*

*Condition or Error*

Replicator (extractor) is unable to stay online and extract an event. Error logs consistently show a stack trace similar to the following:

```
2020/07/24 15:06:14.637 | Event extraction failed
2020/07/24 15:06:14.637 | com.continuent.tungsten.replicator.extractor.ExtractorException: Unexpected failure
» while extracting event myhost-db-04.qa.mydomain.local (1334)
2020/07/24 15:06:14.637 | at com.continuent.tungsten.replicator.extractor.mysql.MySQLExtractor.extractEvent(Unknown Source)
2020/07/24 15:06:14.637 | at com.continuent.tungsten.replicator.extractor.mysql.MySQLExtractor.extract(Unknown Source)
2020/07/24 15:06:14.637 | at com.continuent.tungsten.replicator.extractor.ExtractorWrapper.extract(Unknown Source)
2020/07/24 15:06:14.637 | at com.continuent.tungsten.replicator.extractor.ExtractorWrapper.extract(Unknown Source)
2020/07/24 15:06:14.637 | at com.continuent.tungsten.replicator.pipeline.SingleThreadStageTask.runTask(Unknown Source)
2020/07/24 15:06:14.637 | at com.continuent.tungsten.replicator.pipeline.SingleThreadStageTask.run(Unknown Source)
2020/07/24 15:06:14.637 | at java.lang.Thread.run(Thread.java:748)
2020/07/24 15:06:14.637 | Caused by: java.lang.IndexOutOfBoundsException
2020/07/24 15:06:14.637 | at java.io.DataInputStream.readFully(DataInputStream.java:192)
2020/07/24 15:06:14.637 | at com.continuent.tungsten.common.io.BufferedFileDataInput.readFully(Unknown Source)
2020/07/24 15:06:14.637 | at com.continuent.tungsten.replicator.extractor.mysql.BinlogReader.read(Unknown Source)
2020/07/24 15:06:14.637 | at com.continuent.tungsten.replicator.extractor.mysql.LogEvent.readDataFromBinlog(Unknown Source)
2020/07/24 15:06:14.637 | at com.continuent.tungsten.replicator.extractor.mysql.LogEvent.readLogEvent(Unknown Source)
2020/07/24 15:06:14.637 | at com.continuent.tungsten.replicator.extractor.mysql.MySQLExtractor.processFile(Unknown Source)
2020/07/24 15:06:14.637 | ... 7 more
```

*Causes*

- You could be hitting a MySQL bug where the binlog is over-writing itself due to periods in the log-bin my.cnf entry. See https://bugs.mysql.com/bug.php?id=75507 for more details.

  Example of my.cnf entry that may trigger this bug:

```
log-bin = /data/mysql/myhost-db-04.qa.mydomain.local.com-bin
```

*Rectifications*

- Replace the dots with hyphens and restart MySQL

  Example of a fixed my.cnf entry:

```
log-bin = /data/mysql/myhost-db-04-qa-mydomain-local-com-bin
```

  Adjusting the binlog pattern within MySQL may also require a configuration change to the replicator if the pattern is changed after installation.

  To do this, add the `repl-datasource-log-pattern` to your configuration and issue tpm update

## C.3.8. Attempt to write new log record with equal or lower fragno: seqno=3 previous stored fragno=32767 attempted new fragno=-32768

*Last Updated: 2016-05-18*

*Condition or Error*

The number of fragments in a single transaction has been exceeded.

*Causes*

- The maximum number of fragments within a single transaction within the network protocol is limited to 32768. If there is a very large transaction that exceeds this number of fragments, the replicator can stop and be unable to continue. The total transaction size is a combination of the fragment size (default is 1,000,000 bytes, or 1MB), and this maximum number (approximately 32GB).

*Rectifications*

- It is not possible to change the number of fragments in a single transaction, but the size of each fragment can be increased to handle much larger single transactions. To change the fragment size, configure the `replicator.extractor.dbms.transaction_frag_size` parameter. For example, by doubling the size, a transaction of 64GB could be handled:

```
replicator.extractor.dbms.transaction_frag_size=2000000
```

  If you change the fragment size in this way, the service on the extractor must be reset so that the transaction can be reprocessed and the binary log is parsed again. You can reset the service by using the trepctl reset command.

## C.3.9. The session variable SQL_MODE when set to include ALLOW_INVALID_DATES does not apply statements correctly on the Replica.

*Last Updated: 2013-07-17*

*Condition or Error*

Replication fails due to an incorrect SQL mode, `INVALID_DATES` being applied for a specific transaction.

*Causes*

- Due to a problem with the code, the `SQL_MODE` variable in MySQL when set to include `ALLOW_INVALID_DATES` would be identified incorrectly as `INVALID_DATES` from the information in the binary log.

*Rectifications*

- In affected versions, these statements can be bypassed by explicitly ignoring that value in the event by editing `tungsten-replicator/conf/replicator.properties` to include the following property line:

```
replicator.applier.dbms.ignoreSessionVars=autocommit|INVALID_DATES
```

## C.3.10. Replicator runs out of memory

*Last Updated: 2016-05-18*

*Condition or Error*

The replicator runs out of memory, triggers a stack trace indicator a memory condition, or the replicator fails to extract the transaction information from the MySQL binary log.

*Causes*

- The replicator operates by extracting (or applying) an entire transaction. This means that when extracting data from the binary log, and writing that to THL, or extracting from the THL in preparation for applying to the target, the entire transaction, or an entire statement within a multi-statement transaction, must be held in memory.

  In the event of a very large transaction having to be extracted, this can cause a problem with the memory configuration. The actual configuration of how much memory is used is determined through a combination of the number of fragments, the size of the internal buffer used to store those fragments, and the overall fragment size.

*Rectifications*

- Although you can increase the overall memory allocated to the replicator, changing the internal sizes used can also improve the performance and ability to extract data.

  First, try reducing the size of the buffer ( `replicator.global.buffer.size` ) used to hold the transaction fragments. The default for this value is 10, but reducing this to 5 or less will ease the required memory:

```
replicator.global.buffer.size=10
```

  Altering the size of each fragment can also help, as it reduces the memory required to hold the data before it is written to disk and sent out over the network to Replica replicators. Reducing the fragment size will reduce the memory footprint. The size is controlled by the `replicator.extractor.dbms.transaction_frag_size` parameter:

```
replicator.extractor.dbms.transaction_frag_size=1000000
```

Note that if you change the fragment size, you may need to reset the service on the extractor so that the binary log is parsed again. You can reset the service by using the trepctl reset command.

# C.4. Known Issues

## C.4.1. Triggers

Tungsten Replicator does not automatically shut off triggers on Replicas. This can create problems on Replicas as the trigger will run twice. Typical symptoms are duplicate key errors, though other problems may appear.

There is no simple one-answer-fits-all solution as the behaviour of MySQL and Triggers will differ based on various conditions.

- When using `ROW` Based Binary Logging, MySQL will log all data changes in the binary log, including any data changes performed as a result of a trigger firing
- When using `MIXED` Based Binary Logging...
  - if the Trigger is deemed to be non-deterministic then MySQL will behave based on the `ROW` Based Logging rules and log all data changes, including any data changes performed as a result of a trigger firing.
  - if the Tigger is deemed to be deterministic, then MySQL will behave based on `STATEMENT` Based Logging rules and ONLY log the statement issued by the client and NOT log any changes as a result of the trigger firing

The mixed behaviour outlined above presents challenges for Tungsten Replicator because MySQL does not flag transactions as being the result of a trigger firing or a client application. Therefore, it is not possible for the replicator to make a decision either.

This means, that if you are running with `MIXED` Based Binary Logging enabled, then there may be times when you would want the triggers on the target to fire, and times when you don't. Therefore the recommendations are as follows:

Tungsten Clustering Deployments

- Switch to `ROW` Based Binary Logging, and either
  - Implement the `is_Primary()` function outlined below, or
  - Use the `replicate.ignore` filter to ignore data changes to tables altered by Triggers (ONLY suitable if the filtered tables are solely managed by the Trigger)

Tungsten Replicator Deployments

- If source instance is running in `ROW` Based Binary Logging mode
  - Drop triggers on target. This is practical in fan-in topologies for reporting or other cases where you do not need to failover to the Replica at a later time. Optionally also implement the `dropddl.js` JavaScript filter (Available in Tungsten Replicator v6.1.2 onwards) to prevent CREATE/DROP TRIGGER DDL being replicated, or
  - Implement the `is_Primary()` function outlined below, or
  - Use the `replicate.ignore` filter to ignore data changes to tables altered by Triggers (ONLY suitable if the filtered tables are solely managed by the Trigger)
- If source instance is running in `MIXED` Based Binary Logging mode
  - Use the `replicate.ignore` filter to ignore data changes to tables altered by Triggers (ONLY suitable if the filtered tables are solely managed by the Trigger), or
  - Switch to `ROW` Based Binary Logging and follow recommendations above

The `is_Primary()` approach is simple to implement. First, create a function like the following that returns false if we are using the Tungsten user, as would be the case on a Replica.

```
create function is_Primary()
    returns boolean
    deterministic
    return if(substring_index(user(),'@',1) != 'tungsten',true, false);
```

Next add this to triggers that should not run on the Replica, as shown in the next example. This suppresses trigger action to insert into table bar except on the Primary.

```
delimiter //
create trigger foo_insert after insert on foo
```

```
  for each row begin
    if is_Primary() then
      insert into bar set id=NEW.id;
    end if;
  end;
//
```

As long as applications do not use the Tungsten account on the Primary, the preceding approach will be sufficient to suppress trigger operation.

Alternatively, if you are implementing the `is_Primary()` within a clustering deployment, you could check the database read_only parameter. In a clustered deployment, the Replica databases will be in read_only mode and therefore the trigger could be coded to only fire when the database read_only mode is OFF

# C.5. Troubleshooting Timeouts

# C.6. Troubleshooting Backups

- Operating system command failed

  Backup directory does not exist.

  ```
  ...
  INFO | jvm 1 | 2013/05/21 09:36:47 | Process timed out: false
  INFO | jvm 1 | 2013/05/21 09:36:47 | Process exception null
  INFO | jvm 1 | 2013/05/21 09:36:47 | Process stderr: Error: »
      The directory '/opt/continuent/backups/xtrabackup' is not writeable
  ...
  ```

- Backup Retention

  The number of backups retained is set in the backup retention field. To set it at installation time, use the `--backup-retention=N` option where N is the number of backups to retain.

  You can check the number of currently retained backups by looking at the replicator.properties file and searching for the following property:

  ```
  replicator.storage.agent.fs.retention=3
  ```

  The default is 3 backups retained at any given time.

# C.7. Running Out of Diskspace

```
...
pendingError           : Event application failed: seqno=156847 »
    fragno=0 message=Unable to store event: seqno=156847
pendingErrorCode       : NONE
pendingErrorEventId    : mysql-bin.000025:0000000024735754;0
pendingErrorSeqno      : 156847
pendingExceptionMessage: Unable to store event: seqno=156847
...
```

The above indicates that the THL information could not be stored on disk. To recover from this error, make space available on the disk, or move the THL files to a different device with more space, then set the replicator service online again.

For more information on moving THL files to a different disk, see Section D.1.5.3, “Moving the THL File Location”; for information on moving the backup file location, see Section D.1.1.4, “Relocating Backup Storage”.

# C.8. Troubleshooting SSH and tpm

When executing tpm, ssh is used to connect and install the software on other hosts in the cluster. If this fails, and the public key information is correct, there are a number of operations and settings that can be checked. Ensure that you have followed the Section B.3.2.2, “SSH Configuration” instructions.

- The most likely representation of this error will be when executing tpm during a deployment:

  ```
  Error:
  ##################################################################

  Validation failed
  ##################################################################
  ```

```
#####################################################################
Errors for host1
#####################################################################
ERROR>>host1>>Unable to SSH to host1 as root. (SSHLoginCheck)
Ensure that the host is running and that you can login as root via SSH using key authentication
tungsten-configure.log shows:
2012-05-23T11:10:37+02:00 DEBUG>>Execute `whoami` on host1 as root
2012-05-23T11:10:38+02:00 DEBUG>>RC: 0, Result: stdin: is not a tty
```

Try running the following command:

```
shell> ssh tungsten@host1 sudo whoami
```

If the SSH and sudo configurations have been configured correctly, it should return `root`. Any other value indicates a failure to configure the prerequisites properly.

- Check that none of the profile scripts (`.profile`, `.bash_profile`, `.bashrc`, etc.) do not contain a call to mesg n. This may fool the non-interactive ssh call; the call to this command should be changed to only be executed on interactive shells:

```
if `tty -s`; then
   mesg n
fi
```

- Check that firewalls and/or antivirus software are not blocking or preventing connectivity on port 22.

  If ssh has been enabled on a non-standard port, use the `--net-ssh-option=port` [368] option to specify the alternative port.

- Make sure that the user specified in the `--user` [429] to tpm is allowed to connect to your cluster nodes.

# C.9. Troubleshooting Data Differences

It can sometimes become necessary to identify table and data differences due to unexpected behaviour or failures. There are a number of third party tools that can help identify and fix however a lot of them assume native replication is in place, the following explains the recommended methods for troubleshooting a Tungsten Environment based on MySQL as the source and target technologies.

## C.9.1. Identify Structural Differences

If you suspect that there are differences to a table structure, a simple method to resolve this will be to compare schema DDL.

Extract DDL on the Primary node, specifying the schema in place of {DB}:

```
shell> mysqldump -u root -p --no-data -h localhost --databases {DB} >Primary.sql
```

Repeat the same on the Replica node:

```
shell> mysqldump -u root -p --no-data -h localhost --databases {DB} >Replica.sql
```

Now, using diff, you can compare the results

```
shell> diff Primary.sql Replica.sql
```

Using the output of diff, you can then craft the necessary SQL statements to re-align your structure

## C.9.2. Identify Data Differences

It is possible to use pt-table-checksum from the Percona Toolkit to identify data differences, providing you use the syntax described below for bypassing the native replication checks. First of all, it is advisable to familiarise yourself with the product by reading through the providers own documentation here:

https://www.percona.com/doc/percona-toolkit/2.2/pt-table-checksum.html

Once you are ready, ensure you install the latest version to the persona toolkit on all nodes, next execute the following on the Primary node:

```
shell> pt-table-checksum --set-vars innodb_lock_wait_timeout=500 \
--recursion-method=none \
--ignore-databases=mysql \
--ignore-databases-regex=tungsten* \
h=localhost,u=tungsten,p=secret
```

On first run, this will create a database called percona, and within that database a table called checksums. The process will gather checksum information on every table in every database excluding the mysql and tungsten related schemas. You can now execute the following SQL Statement on the Replica to identify tables with data differences:

```
SELECT db, tbl, SUM(this_cnt) AS total_rows, COUNT(*) AS chunks
FROM percona.checksums
WHERE (
 master_cnt <> this_cnt
 OR master_crc <> this_crc
 OR ISNULL(master_crc) <> ISNULL(this_crc))
GROUP BY db, tbl;
```

This `SELECT` will return any tables that it detects are different, it won't show you the differences, or indeed how many, this is just a basic check. To identify and fix the changes, you could use pt-table-sync, however this product would by default assume native replication and also try and fix the problems for you. In a tungsten environment this would not be recommended, however by using the `--print` switch you can gather the SQL needed to be executed to fix the mistakes. You should run this, and review the output to determine whether you want to manually patch the data together or consider using tungsten_provision_slave to retrovision a node in the case of large quantities of differences.

To use pt-table-sync, first identify the tables with differences on each Replica, in this example, the `SELECT` statement above identified that there was a data difference on the departments table within the employees database on db2. Execute the pt-table-sync script on the Primary, passing in the database name, table name and the Replica host that the difference exists on:

```
shell> pt-table-sync --databases employees --tables departments --print h=db1,u=tungsten,p=secret,P=13306 h=db2
```

The first `h=` option should be the Primary, also the node you run the script from, the second `h=` option relates to the Replica that the difference exists on. Executing the script will output SQL statements that can be used to patch the data, for example the above statement produces the following output:

```
UPDATE `employees`.`departments`
SET `dept_name`='Sales'
WHERE `dept_no`='d007'
LIMIT 1
/*percona-toolkit src_db:employees src_tbl:departments src_dsn:P=13306,h=db1,p=...,u=tungsten
dst_db:employees dst_tbl:departments dst_dsn:P=13306,h=db2,p=...,u=tungsten
lock:0 transaction:1 changing_src:0 replicate:0 bidirectional:0 pid:24524 user:tungsten host:db1*/;
```

The `UPDATE` statements could now be issued directly on the Replica to correct the problem.

### Warning

> Generally, changing data directly on a Replica is not recommended, but every environment is different. before making any changes like this *always* ensure you have a FULL backup, and it would be recommended to shun the Replica node (if in a clustered environment) before making any changes so as not to cause any potential interruption to connected clients

## C.10. Comparing Table Data

The Percona Toolkit includes a tool called pt-table-checksum that enables you to compare databases on different databases using a checksum comparison. This can be executed by running the checksum generation process on the Primary:

```
shell> pt-table-checksum --set-vars innodb_lock_wait_timeout=500 \
    --recursion-method=none \
    --ignore-databases=mysql \
    --ignore-databases-regex=tungsten* \
    h=localhost,u=tungsten,p=secret
```

Using MySQL, the following statement must then be executed to check the checksums generated on the Primary:

```
mysql> <userinput>SELECT db, tbl, SUM(this_cnt) AS total_rows, COUNT(*) AS chunks \
    FROM percona.checksums WHERE ( master_cnt <> this_cnt OR master_crc \
    <> this_crc OR ISNULL(master_crc) <> ISNULL(this_crc)) GROUP BY db, tbl;</userinput>
```

Any differences will be reported and will need to manually corrected.

## C.11. Troubleshooting Memory Usage

# Appendix D. Files, Directories, and Environment

## D.1. The Tungsten Cluster Install Directory

Any Tungsten Cluster™ installation creates an installation directory that contains the software and the additional directories where active information, such as the transaction history log and backup data is stored. A sample of the directory is shown below, and a description of the individual directories is provided in Table D.1, "Continuent Tungsten Directory Structure".

```
shell> ls -al /opt/continuent
total 40
drwxr-xr-x 9 tungsten root     4096 Mar 21 18:47 .
drwxr-xr-x 3 root     root     4096 Mar 21 18:00 ..
drwxrwxr-x 2 tungsten tungsten 4096 Mar 21 18:44 backups
drwxrwxr-x 2 tungsten tungsten 4096 Mar 21 18:47 conf
drwxrwxr-x 3 tungsten tungsten 4096 Mar 21 18:44 relay
drwxrwxr-x 4 tungsten tungsten 4096 Mar 21 18:47 releases
drwxrwxr-x 2 tungsten tungsten 4096 Mar 21 18:47 service_logs
drwxrwxr-x 2 tungsten tungsten 4096 Mar 21 18:47 share
drwxrwxr-x 3 tungsten tungsten 4096 Mar 21 18:44 thl
lrwxrwxrwx 1 tungsten tungsten   62 Mar 21 18:47 tungsten -> /opt/continuent/releases/tungsten-replicator-7.1.2-81_pid31409
```

The directories shown in the table are relative to the installation directory, the recommended location is `/opt/continuent`. For example, the THL files would be located in `/opt/continuent/thl`.

Table D.1. Continuent Tungsten Directory Structure

| Directory | Description |
|---|---|
| `backups` | Default directory for backup file storage |
| `conf` | Configuration directory with a copy of the current and past configurations |
| `relay` | Location for relay logs if relay logs have been enabled. |
| `releases` | Contains one or more active installations of the Continuent Tungsten software, referenced according to the version number and active process ID. |
| `service-logs` | Logging information for the active installation |
| `share` | Active installation information, including the active JAR for the MySQL connection |
| `thl` | The Transaction History Log files, stored in a directory named after each active service. |
| `tungsten` | Symbolic link to the currently active release in `releases`. |

Some advice for the contents of specific directories within the main installation directory are described in the following sections.

### D.1.1. The `backups` Directory

The `backups` directory is the default location for the data and metadata from any backup performed manually or automatically by Tungsten Cluster™. The backup data and metadata for each backup will be stored in this directory.

An example of the directory content is shown below:

```
shell> ls -al /opt/continuent/backups/
total 130788
drwxrwxr-x 2 tungsten tungsten      4096 Apr  4 16:09 .
drwxrwxr-x 3 tungsten tungsten      4096 Apr  4 11:51 ..
-rw-r--r-- 1 tungsten tungsten        71 Apr  4 16:09 storage.index
-rw-r--r-- 1 tungsten tungsten 133907646 Apr  4 16:09 store-0000000001-mysqldump_2013-04-04_16-08_42.sql.gz
-rw-r--r-- 1 tungsten tungsten       317 Apr  4 16:09 store-0000000001.properties
```

The `storage.index` contains the backup file index information. The actual backup data is stored in the GZipped file. The properties of the backup file, including the tool used to create the backup, and the checksum information, are location in the corresponding `.properties` file. Note that each backup and property file is uniquely numbered so that you can identify and restore a specific backup.

Different backups scripts and methods may place their backup information in a separate subdirectory. For example, xtrabackup stores backup data into `/opt/continuent/backups/xtrabackup`.

#### D.1.1.1. Automatically Deleting Backup Files

The Tungsten Replicator will automatically remove old backup files. This is controlled by the `--repl-backup-retention` [397] setting and defaults to 3. Use the tpm update command to modify this setting. Following the successful creation of a new backup, the number of backups will be

compared to the retention value. Any excess backups will be removed from the `/opt/continuent/backups` directory or whatever directory is configured for `--repl-backup-directory` [396].

The backup retention will only remove files starting with `store`. If you are using a backup method that creates additional information then those files may not be fully removed until the next backup process begins. This includes xtrabackup-full, xtrabackup-incremental and any snapshot based backup methods. You may manually clean these excess files if space is needed before the next backup method. If you delete information associated with an existing backup, any attempts to restore it will fail.

## D.1.1.2. Manually Deleting Backup Files

If you no longer need one or more backup files, you can delete the files from the filesystem. You must delete both the SQL data, and the corresponding properties file. For example, from the following directory:

```
shell> ls -al /opt/continuent/backups
total 764708
drwxrwxr-x 2 tungsten tungsten      4096 Apr 16 13:57 .
drwxrwxr-x 3 tungsten tungsten      4096 Apr 16 13:54 ..
-rw-r--r-- 1 tungsten tungsten        71 Apr 16 13:56 storage.index
-rw-r--r-- 1 tungsten tungsten    517170 Apr 15 18:02 store-0000000004-mysqldump-1332463738918435527.sql
-rw-r--r-- 1 tungsten tungsten       311 Apr 15 18:02 store-0000000004.properties
-rw-r--r-- 1 tungsten tungsten    517170 Apr 15 18:06 store-0000000005-mysqldump-2284057977980000458.sql
-rw-r--r-- 1 tungsten tungsten       310 Apr 15 18:06 store-0000000005.properties
-rw-r--r-- 1 tungsten tungsten 781991444 Apr 16 13:57 store-0000000006-mysqldump-3081853249977885370.sql
-rw-r--r-- 1 tungsten tungsten       314 Apr 16 13:57 store-0000000006.properties
```

To delete the backup files for index 4:

```
shell> rm /opt/continuent/backups/alpha/store-0000000004*
```

See the information in Section D.1.1.3, “Copying Backup Files” about additional files related to a single backup. There may be additional files associated with the backup that you will need to manually remove.

### Warning

> Removing a backup should only be performed if you know that the backup is safe to be removed and will not be required. If the backup data is required, copy the backup files from the backup directory before deleting the files in the backup directory to make space.

## D.1.1.3. Copying Backup Files

The files created during any backup can copied to another directory or system using any suitable means. Once the backup has been completed, the files will not be modified or updated and are therefore safe to be moved or actively copied to another location without fear of corruption of the backup information.

There are multiple files associated with each backup. The number of files will depend on the backup method that was used. All backups will use at least two files in the `/opt/continuent/backups` directory.

```
shell> cd /opt/continuent/backups
shell> scp store-[0]*6[\.-]* host3:$PWD/
store-0000000001-full_xtrabackup_2014-08-16_15-44_86                  100%   70     0.1KB/s   00:00
store-0000000001.properties                                           100%  314     0.3KB/s   00:00
```

### Note

> Check the ownership of files if you have trouble transferring files or restoring the backup. They should be owned by the Tungsten system user to ensure proper operation.

If xtrabackup-full method was used, you must transfer the corresponding directory from `/opt/continuent/backups/xtrabackup`. In this example that would be `/opt/continuent/backups/xtrabackup/full_xtrabackup_2014-08-16_15-44_86`.

```
shell> cd /opt/continuent/backups/xtrabackup
shell> rsync -aze ssh full_xtrabackup_2014-08-16_15-44_86 host3:$PWD/
```

If the xtrabackup-incremental method was used, you must transfer multiple directories. In addition to the corresponding directory from `/opt/continuent/backups/xtrabackup` you must transfer all xtrabackup-incremental directories since the most recent xtrabackup-full backup and then transfer that xtrabackup-full directory. See the example below for further explanation :

```
shell> ls -altr /opt/continuent/backups/xtrabackup/
total 32
drwxr-xr-x 7 tungsten tungsten 4096 Oct 16 20:55 incr_xtrabackup_2014-10-16_20-55_73
drwxr-xr-x 7 tungsten tungsten 4096 Oct 17 20:55 full_xtrabackup_2014-10-17_20-55_1
drwxr-xr-x 7 tungsten tungsten 4096 Oct 18 20:55 incr_xtrabackup_2014-10-18_20-55_38
drwxr-xr-x 7 tungsten tungsten 4096 Oct 19 20:57 incr_xtrabackup_2014-10-19_20-57_76
```

```
drwxr-xr-x 7 tungsten tungsten 4096 Oct 20 20:58 full_xtrabackup_2014-10-20_20-57_41
drwxr-xr-x 8 tungsten tungsten 4096 Oct 21 20:58 .
drwxr-xr-x 7 tungsten tungsten 4096 Oct 21 20:58 incr_xtrabackup_2014-10-21_20-58_97
drwxrwxr-x 3 tungsten tungsten 4096 Oct 21 20:58 ..
```

In this example there are two instances of xtrabackup-full backups and four xtrabackup-incremental backups.

- To restore either of the xtrabackup-full backups then they would be copied to the target host on their own.
- To restore `incr_xtrabackup_2014-10-21_20-58_97`, it must be copied along with `full_xtrabackup_2014-10-20_20-57_41`.
- To restore `incr_xtrabackup_2014-10-19_20-57_76`, it must be copied along with `incr_xtrabackup_2014-10-18_20-55_38` and `full_xtrabackup_2014-10-17_20-55_1`.

## D.1.1.4. Relocating Backup Storage

If the filesystem on which the main installation directory is running out of space and you need to increase the space available for backup files without interrupting the service, you can use symbolic links to relocate the backup information.

### Note

When using an NFS mount point when backing up with xtrabackup , the command must have the necessary access rights and permissions to change the ownership of files within the mounted directory. Failure to update the permissions and ownership will cause the xtrabackup command to fail. The following settings should be made on the directory:

- Ensure the `no_root_squash` option on the NFS export is not set.
- Change the group and owner of the mount point to the `tungsten` user and `mysql` group:

  ```
  shell> chown tungsten /mnt/backups
  shell> chgrp mysql /mnt/backups
  ```

  Owner and group IDs on NFS directories must match across all the hosts using the NFS mount point. Inconsistencies in the owner and group IDs may lead to backup failures.

- Change the permissions to permit at least owner and group modifications::

  ```
  shell> chmod 770 /mnt/backups
  ```

- Mount the directory:

  ```
  shell> mount host1:/exports/backups /mnt/backups
  ```

The backup directory can be changed using two different methods:

- Section D.1.1.4.1, “Relocating Backup Storage using Symbolic Links”
- Section D.1.1.4.2, “Relocating Backup Storage using Configuration Changes”

### D.1.1.4.1. Relocating Backup Storage using Symbolic Links

To relocate the backup directory using symbolic links:

1. Ensure that no active backup is taking place of the current host. Your service does not need to be offline to complete this operation.
2. Create a new directory, or attach a new filesystem and location on which the backups will be located. You can use a directory on another filesystem or connect to a SAN, NFS or other filesystem where the new directory will be located. For example:

   ```
   shell> mkdir /mnt/backupdata/continuent
   ```

3. *Optional*

   Copy the existing backup directory to the new directory location. For example:

   ```
   shell> rsync -r /opt/continuent/backups/* /mnt/backupdata/continuent/
   ```

4. Move the existing directory to a temporary location:

   ```
   shell> mv /opt/continuent/backups /opt/continuent/old-backups
   ```

5. Create a symbolic link from the new directory to the original directory location:

```
shell> ln -s /mnt/backupdata/continuent /opt/continuent/backups
```

The backup directory has now been moved. If you want to verify that the new backup directory is working, you can optionally run a backup and ensure that the backup process completes correctly.

#### D.1.1.4.2. Relocating Backup Storage using Configuration Changes

To relocate the backup directory by reconfiguration:

1. Ensure that no active backup is taking place of the current host. Your service does not need to be offline to complete this operation.
2. Create a new directory, or attach a new filesystem and location on which the backups will be located. You can use a directory on another filesystem or connect to a SAN, NFS or other filesystem where the new directory will be located. For example:

   ```
   shell> mkdir /mnt/backupdata/continuent
   ```

3. *Optional*

   Copy the existing backup directory to the new directory location. For example:

   ```
   shell> rsync -r /opt/continuent/backups/* /mnt/backupdata/continuent/
   ```

4. Following the directions for tpm update to apply the `--backup-directory=/mnt/backupdata/continuent` [396] setting.

The backup directory has now been moved. If you want to verify that the new backup directory is working, you can optionally run a backup and ensure that the backup process completes correctly.

## D.1.2. The `releases` Directory

The `releases` directory contains a copy of each installed release. As new versions are installed and updated (through tpm update), a new directory is created with the corresponding version of the software.

For example, a number of releases are listed below:

```
shell> ll /opt/continuent/releases/
total 20
drwxr-xr-x  5 tungsten mysql 4096 May 23 16:19 ./
drwxr-xr-x  9 tungsten mysql 4096 May 23 16:19 ../
drwxr-xr-x 10 tungsten mysql 4096 May 23 16:19 tungsten-replicator-7.1.2-81_pid16184/
drwxr-xr-x 10 tungsten mysql 4096 May 23 16:19 tungsten-replicator-7.1.2-81_pid14577/
drwxr-xr-x 10 tungsten mysql 4096 May 23 16:19 tungsten-replicator-7.1.2-81_pid23747/
drwxr-xr-x 10 tungsten mysql 4096 May 23 16:19 tungsten-replicator-7.1.2-81_pid24978/
```

The latest release currently in use can be determined by checking the symbolic link, `tungsten` within the installation directory. For example:

```
shell> ll /opt/continuent
total 40
drwxr-xr-x 9 tungsten mysql 4096 May 23 16:19 ./
drwxr-xr-x 3 root     root  4096 Apr 29 16:09 ../
drwxr-xr-x 2 tungsten mysql 4096 May 30 13:27 backups/
drwxr-xr-x 2 tungsten mysql 4096 May 23 16:19 conf/
drwxr-xr-x 3 tungsten mysql 4096 May 10 19:09 relay/
drwxr-xr-x 5 tungsten mysql 4096 May 23 16:19 releases/
drwxr-xr-x 2 tungsten mysql 4096 May 10 19:09 service_logs/
drwxr-xr-x 2 tungsten mysql 4096 May 23 16:18 share/
drwxr-xr-x 3 tungsten mysql 4096 May 10 19:09 thl/
lrwxrwxrwx 1 tungsten mysql   63 May 23 16:19 tungsten -> /opt/continuent/releases/tungsten-replicator-7.1.2-81_pid24978/
```

If multiple services are running on the host, search for `.pid` files within the installation directory to determine which release directories are currently in use by an active service:

```
shell> find /opt/continuent -name "*.pid"
/opt/continuent/releases/tungsten-replicator-7.1.2-81_pid24978/tungsten-replicator/var/treplicator.pid
/opt/continuent/releases/tungsten-replicator-7.1.2-81_pid24978/tungsten-connector/var/tconnector.pid
/opt/continuent/releases/tungsten-replicator-7.1.2-81_pid24978/tungsten-manager/var/tmanager.pid
```

Directories within the `releases` directory that are no longer being used can be safely removed.

## D.1.3. The `service_logs` Directory

The `service_logs` directory contains links to the log files for the currently active release. The directory contains the following links:

- `trepsvc.log` — a link to the Tungsten Replicator log.

## D.1.4. The share Directory

The share directory contains information that is shared among all installed releases and instances of Tungsten Cluster. Unlike other directories, the share directory is not overwritten or replaced during installation or update using tpm. This means that the directory can be used to hold information, such as filter configurations, without the contents being removed when the installation is updated.

## D.1.5. The thl Directory

The transaction history log (THL) retains a copy of the SQL statements from each Primary host, and it is the information within the THL that is transferred between hosts and applied to the database. The THL information is written to disk and stored in the thl directory:

```
shell> ls -al /opt/continuent/thl/alpha/
total 2291984
drwxrwxr-x 2 tungsten tungsten      4096 Apr 16 13:44 .
drwxrwxr-x 3 tungsten tungsten      4096 Apr 15 15:53 ..
-rw-r--r-- 1 tungsten tungsten         0 Apr 15 15:53 disklog.lck
-rw-r--r-- 1 tungsten tungsten 100137585 Apr 15 18:13 thl.data.0000000001
-rw-r--r-- 1 tungsten tungsten 100134069 Apr 15 18:18 thl.data.0000000002
-rw-r--r-- 1 tungsten tungsten 100859685 Apr 15 18:26 thl.data.0000000003
-rw-r--r-- 1 tungsten tungsten 100515215 Apr 15 18:28 thl.data.0000000004
-rw-r--r-- 1 tungsten tungsten 100180770 Apr 15 18:31 thl.data.0000000005
-rw-r--r-- 1 tungsten tungsten 100453094 Apr 15 18:34 thl.data.0000000006
-rw-r--r-- 1 tungsten tungsten 100379260 Apr 15 18:35 thl.data.0000000007
-rw-r--r-- 1 tungsten tungsten 100294561 Apr 16 12:21 thl.data.0000000008
-rw-r--r-- 1 tungsten tungsten 100133258 Apr 16 12:24 thl.data.0000000009
-rw-r--r-- 1 tungsten tungsten 100293278 Apr 16 12:32 thl.data.0000000010
-rw-r--r-- 1 tungsten tungsten 100819317 Apr 16 12:34 thl.data.0000000011
-rw-r--r-- 1 tungsten tungsten 100250972 Apr 16 12:35 thl.data.0000000012
-rw-r--r-- 1 tungsten tungsten 100337285 Apr 16 12:37 thl.data.0000000013
-rw-r--r-- 1 tungsten tungsten 100535387 Apr 16 12:38 thl.data.0000000014
-rw-r--r-- 1 tungsten tungsten 100378358 Apr 16 12:40 thl.data.0000000015
-rw-r--r-- 1 tungsten tungsten 100198421 Apr 16 13:32 thl.data.0000000016
-rw-r--r-- 1 tungsten tungsten 100136955 Apr 16 13:34 thl.data.0000000017
-rw-r--r-- 1 tungsten tungsten 100490927 Apr 16 13:41 thl.data.0000000018
-rw-r--r-- 1 tungsten tungsten 100684346 Apr 16 13:41 thl.data.0000000019
-rw-r--r-- 1 tungsten tungsten 100225119 Apr 16 13:42 thl.data.0000000020
-rw-r--r-- 1 tungsten tungsten 100390819 Apr 16 13:43 thl.data.0000000021
-rw-r--r-- 1 tungsten tungsten 100418115 Apr 16 13:43 thl.data.0000000022
-rw-r--r-- 1 tungsten tungsten 100388812 Apr 16 13:44 thl.data.0000000023
-rw-r--r-- 1 tungsten tungsten  38275509 Apr 16 13:47 thl.data.0000000024
```

THL files are created on both the Primary and Replicas within the cluster. THL data can be examined using the thl command.

The THL is written into individual files, which are by default, no more than 1 GByte in size each. From the listing above, you can see that each file has a unique file index number. A new file is created when the file size limit is reached, and given the next THL log file number. To determine the sequence number that is stored within log, use the thl command:

```
shell> thl index
LogIndexEntry thl.data.0000000001(0:106)
LogIndexEntry thl.data.0000000002(107:203)
LogIndexEntry thl.data.0000000003(204:367)
LogIndexEntry thl.data.0000000004(368:464)
LogIndexEntry thl.data.0000000005(465:561)
LogIndexEntry thl.data.0000000006(562:658)
LogIndexEntry thl.data.0000000007(659:755)
LogIndexEntry thl.data.0000000008(756:1251)
LogIndexEntry thl.data.0000000009(1252:1348)
LogIndexEntry thl.data.0000000010(1349:1511)
LogIndexEntry thl.data.0000000011(1512:1609)
LogIndexEntry thl.data.0000000012(1610:1706)
LogIndexEntry thl.data.0000000013(1707:1803)
LogIndexEntry thl.data.0000000014(1804:1900)
LogIndexEntry thl.data.0000000015(1901:1997)
LogIndexEntry thl.data.0000000016(1998:2493)
LogIndexEntry thl.data.0000000017(2494:2590)
LogIndexEntry thl.data.0000000018(2591:2754)
LogIndexEntry thl.data.0000000019(2755:2851)
LogIndexEntry thl.data.0000000020(2852:2948)
LogIndexEntry thl.data.0000000021(2949:3045)
LogIndexEntry thl.data.0000000022(3046:3142)
LogIndexEntry thl.data.0000000023(3143:3239)
LogIndexEntry thl.data.0000000024(3240:3672)
```

The THL files are retained for seven days by default, although this parameter is configurable. Due to the nature and potential size required to store the information for the THL, you should monitor the disk space and usage.

The purge is continuous and is based on the date the log file was written. Each time the replicator finishes the current THL log file, it checks for files that have exceeded the defined retention configuration and spawns a job within the replicator to delete files older than the retention policy. Old files are only removed when the current THL log file rotates.

### D.1.5.1. Purging THL Log Information on a Replica

#### Warning

> Purging the THL on a Replica node can potentially remove information that has not yet been applied to the database. Please check and ensure that the THL data that you are purging has been applied to the database before continuing.

The THL files can be explicitly purged to recover disk space, but you should ensure that the currently applied sequence no to the database is not purged, and that additional hosts are not reading the THL information.

To purge the logs on a Replica node:

1. Determine the highest sequence number from the THL that you want to delete. To purge the logs up until the latest sequence number, you can use trepctl to determine the highest applied sequence number:

```
shell> trepctl services
Processing services command...
NAME              VALUE
----              -----
appliedLastSeqno: 3672
appliedLatency  : 331.0
role            : slave
serviceName     : alpha
serviceType     : local
started         : true
state           : ONLINE
Finished services command...
```

2. Put the replication service offline using trepctl:

```
shell> trepctl -service alpha offline
```

3. Use the thl command to purge the logs up to the specified transaction sequence number. You will be prompted to confirm the operation:

```
shell> thl purge -high 3670
WARNING: The purge command will break replication if you delete all events or »
   delete events that have not reached all slaves.
Are you sure you wish to delete these events [y/N]?
y
Deleting events where SEQ# <=3670
2013-04-16 14:09:42,384 [ - main] INFO  thl.THLManagerCtrl Transactions deleted
```

4. Put the replication service online using trepctl:

```
shell> trepctl -service alpha online
```

You can now check the current THL file information:

```
shell> thl index
LogIndexEntry thl.data.0000000024(3240:3672)
```

For more information on purging events using thl, see Section 8.19.6, “thl purge Command”.

### D.1.5.2. Purging THL Log Information on a Primary

#### Warning

> Purging the THL on a Primary node can potentially remove information that has not yet been applied to the Replica databases. Please check and ensure that the THL data that you are purging has been applied to the database on all Replicas before continuing.

#### Important

> If the situation allows, it may be better to switch the Primary role to a current, up-to-date Replica, then perform the steps to purge THL from a Replica on the old Primary host using Section D.1.5.1, “Purging THL Log Information on a Replica”.

#### Warning

> Follow the below steps with great caution! Failure to follow best practices will result in Replicas unable to apply transactions, forcing a full re-provisioning. For those steps, please see Section 7.6, “Provision or Reprovision a Replica”.

The THL files can be explicitly purged to recover disk space, but you should ensure that the currently applied sequence no to the database is not purged, and that additional hosts are not reading the THL information.

To purge the logs on a Primary node:

1. Determine the highest sequence number from the THL that you want to delete. To purge the logs up until the latest sequence number, you can use trepctl to determine the highest applied sequence number:

```
shell> trepctl services
Processing services command...
NAME              VALUE
----              -----
appliedLastSeqno: 3675
appliedLatency  : 0.835
role            : master
serviceName     : alpha
serviceType     : local
started         : true
state           : ONLINE
Finished services command...
```

2. Put the replication service offline using trepctl:

```
shell> trepctl -service alpha offline
```

3. Use the thl command to purge the logs up to the specified transaction sequence number. You will be prompted to confirm the operation:

```
shell> thl purge -high 3670
WARNING: The purge command will break replication if you delete all events or »
   delete events that have not reached all slaves.
Are you sure you wish to delete these events [y/N]?
y
Deleting events where SEQ# <=3670
2013-04-16 14:09:42,384 [ - main] INFO  thl.THLManagerCtrl Transactions deleted
```

4. Put the replication service online using trepctl:

```
shell> trepctl -service alpha online
```

You can now check the current THL file information:

```
shell> thl index
LogIndexEntry thl.data.0000000024(3240:3672)
```

For more information on purging events using thl, see Section 8.19.6, “thl purge Command”.

## D.1.5.3. Moving the THL File Location

The location of the THL directory where THL files are stored can be changed, either by using a symbolic link or by changing the configuration to point to the new directory:

- Changing the directory location using symbolic links can be used in an emergency if the space on a filesystem has been exhausted. See Section D.1.5.3.1, “Relocating THL Storage using Symbolic Links”
- Changing the directory location through reconfiguration can be used when a permanent change to the THL location is required. See Section D.1.5.3.2, “Relocating THL Storage using Configuration Changes”.t

### D.1.5.3.1. Relocating THL Storage using Symbolic Links

In an emergency, the directory currently holding the THL information, can be moved using symbolic links to relocate the files to a location with more space.

Moving the THL location requires updating the location for a Replica by temporarily setting the Replica offline, updating the THL location, and re-enabling back into the cluster:

1. Put the replication service offline using trepctl:

```
shell> trepctl -service alpha offline
```

2. Create a new directory, or attach a new filesystem and location on which the THL content will be located. You can use a directory on another filesystem or connect to a SAN, NFS or other filesystem where the new directory will be located. For example:

```
shell> mkdir /mnt/data/thl
```

3. Copy the existing THL directory to the new directory location. For example:

```
shell> rsync -r /opt/continuent/thl/* /mnt/data/thl/
```

4. Move the existing directory to a temporary location:

```
shell> mv /opt/continuent/thl /opt/continuent/old-thl
```

5. Create a symbolic link from the new directory to the original directory location:

```
shell> ln -s /mnt/data/thl /opt/continuent/thl
```

6. Put the replication service online using trepctl:

```
shell> trepctl -service alpha online
```

#### D.1.5.3.2. Relocating THL Storage using Configuration Changes

To permanently change the directory currently holding the THL information can by reconfigured to a new directory location.

To update the location for a Replica by temporarily setting the Replica offline, updating the THL location, and re-enabling back into the cluster:

1. Put the replication service offline using trepctl:

```
shell> trepctl -service alpha offline
```

2. Create a new directory, or attach a new filesystem and location on which the THL content will be located. You can use a directory on another filesystem or connect to a SAN, NFS or other filesystem where the new directory will be located. For example:

```
shell> mkdir /mnt/data/thl
```

3. Copy the existing THL directory to the new directory location. For example:

```
shell> rsync -r /opt/continuent/thl/* /mnt/data/thl/
```

4. Change the directory location using tpm to update the configuration for a specific host:

```
shell> tpm update --thl-directory=/mnt/data/thl --host=host1
```

5. Put the replication service online using trepctl:

```
shell> trepctl -service alpha online
```

### D.1.5.4. Changing the THL Retention Times

THL files are by default retained for seven days, but the retention period can be adjusted according the to requirements of the service. Longer times retain the logs for longer, increasing disk space usage while allowing access to the THL information for longer. Shorter logs reduce disk space usage while reducing the amount of log data available.

> **Note**
>
> The files are automatically managed by Tungsten Cluster. Old THL files are deleted only when new data is written to the current files. If there has been no THL activity, the log files remain until new THL information is written.

Use the tpm update command to apply the `--repl-thl-log-retention` [428] setting. The replication service will be restarted on each host with updated retention configuration.

## D.1.6. The `tungsten` Directory

```
shell> ls -l /opt/continuent/tungsten/
total 72
drwxr-xr-x  9 tungsten mysql  4096 May 23 16:18 bristlecone
drwxr-xr-x  6 tungsten mysql  4096 May 23 16:18 cluster-home
drwxr-xr-x  4 tungsten mysql  4096 May 23 16:18 cookbook
-rw-r--r--  1 tungsten mysql   681 May 23 16:18 INSTALL
-rw-r--r--  1 tungsten mysql 19974 May 23 16:18 README.LICENSES
drwxr-xr-x  3 tungsten mysql  4096 May 23 16:18 tools
-rw-r--r--  1 tungsten mysql 19724 May 23 16:18 tungsten.cfg
drwxr-xr-x 11 tungsten mysql  4096 May 23 16:18 tungsten-replicator
```

Table D.2. Continuent Tungsten `tungsten` Sub-Directory Structure

| Directory | Description |
|---|---|
| `bristlecone` | Contains the bristlecone load-testing tools. |

| Directory | Description |
|---|---|
| `cluster-home` | Home directory for the main tools, configuration and libraries of the Tungsten Cluster installation. |
| `cookbook` | Cookbook installation and testing tools. |
| `INSTALL` | Text file describing the basic installation process for Tungsten Cluster |
| `README.LICENSES` | Software license information. |
| `tools` | Directory containing the tools for installing and configuring Tungsten Cluster. |
| `tungsten-replicator` | Installed directory of the Tungsten Replicator installation. |

#### D.1.6.1. The `tungsten-replicator` Directory

This directory holds all of the files, libraries, configuration and other information used to support the installation of Tungsten Manager.

##### D.1.6.1.1. The `tungsten-replicator/lib` Directory

This directory holds library files specific to Tungsten Replicator. When perform patches or extending functionality specifically for Tungsten Replicator, for example when adding JDBC libraries for other databases, the JAR files can be placed into this directory.

##### D.1.6.1.2. The `tungsten-replicator/scripts` Directory

This directory contains scripts used to support Tungsten Replicator operation.

## D.2. Log Files

The replicator generates it's own log files. These log files are all written into their own directory within the installation directory structure. In addition, symbolic links are generated for easier access to light weight logs more suited for general user use.

For example, this is the listing of the default log directory, /opt/continuent/service_logs:

```
mysqldump.log -> /opt/continuent/tungsten/tungsten-replicator/log/mysqldump.log
replicator-user.log -> /opt/continuent/tungsten/tungsten-replicator/log/replicator-user.log
trepsvc.log -> /opt/continuent/tungsten/tungsten-replicator/log/trepsvc.log
xtrabackup.log -> /opt/continuent/tungsten/tungsten-replicator/log/xtrabackup.log
```

As you can see, each log file is a symlink to the user-level log, and the more detailed log, along with logs for backups, if they exist.

## D.3. Environment Variables

- `$CONTINUENT_PROFILES`

  This environment variable is used by tpm as the location for storing the `deploy.cfg` file that is created by tpm during a tpm configure or tpm install operation. For more information, see Section 9.3, “tpm Staging Configuration”.

- `$REPLICATOR_PROFILES`

  When using tpm with Tungsten Replicator, `$REPLICATOR_PROFILES` is used for storing the `deploy.cfg` file during configuration and installation. If `$REPLICATOR_PROFILES` does not exist, then `$CONTINUENT_PROFILES` if it exists. For more information, see Section 9.3, “tpm Staging Configuration”.

- `$CONTINUENT_ROOT`

  The `$CONTINUENT_ROOT` variable is created by the `env.sh` file that is created when installing Tungsten Cluster. When defined, the variable will contain the installation directory of the corresponding Tungsten Cluster installation.

  On hosts where multiple installations have been created, the variable can be used to point to different installations.

# Appendix E. Terminology Reference

Tungsten Cluster involves a number of different terminology that helps define different parts of the product, and specific areas of the output information from different commands. Some of this information is shared across different tools and systems.

This appendix includes a reference to the most common terms and terminology used across Tungsten Cluster.

## E.1. Transaction History Log (THL)

The Transaction History Log (THL) stores transactional data from different data servers in a universal format that is then used to exchange and transfer the information between replicator instances. Because the THL is stored and independently managed from the data servers that it reads and writes, the data can be moved, exchanged, and transmuted during processing.

The THL is created by any replicator service acting as a Primary, where the information is read from the database using the native format, such as the MySQL binary log, or Oracle Change Data Capture (CDC), writing the information to the THL. Once in the THL, the THL data can be exchanged with other processes, including transmission over the network, and then applied to a destination database. Within Tungsten Replicator, this process is handled through the pipeline stages that read and write information between the THL and internal queues.

Information stored in THL is recorded in a series of event records in sequential format. The THL therefore acts as a queue of the transactions. On a replicator reading data from a database, the THL represents the queue of transactions applied on the source database. On a replicator applying that information to a database, the THL represents the list of the transactions to be written. The THL has the following properties:

- THL is a sequential list of events
- THL events are written to a THL file through a single thread (to enforce the sequential nature)
- THL events can be read from individually or sequentially, and multiple threads can read the same THL at the same time
- THL events are immutable; once stored, the contents of the THL are never modified or individually deleted (although entire files may be deleted)
- THL is written to disk without any buffering to prevent software failure causing a problem; the operating system buffers are used.

THL data is stored on disk within the `thl` directory of your Tungsten Replicator installation. The exact location can configured using *`logDir`* parameter of the THL component. A sample directory is shown below:

```
total 710504
-rw-r--r-- 1 tungsten tungsten         0 May  2 10:48 disklog.lck
-rw-r--r-- 1 tungsten tungsten 100042900 Jun  4 10:10 thl.data.0000000013
-rw-rw-r-- 1 tungsten tungsten 101025311 Jun  4 11:41 thl.data.0000000014
-rw-rw-r-- 1 tungsten tungsten 100441159 Jun  4 11:43 thl.data.0000000015
-rw-rw-r-- 1 tungsten tungsten 100898492 Jun  4 11:44 thl.data.0000000016
-rw-rw-r-- 1 tungsten tungsten 100305613 Jun  4 11:44 thl.data.0000000017
-rw-rw-r-- 1 tungsten tungsten 100035516 Jun  4 11:44 thl.data.0000000018
-rw-rw-r-- 1 tungsten tungsten 101690969 Jun  4 11:45 thl.data.0000000019
-rw-rw-r-- 1 tungsten tungsten  23086641 Jun  5 21:55 thl.data.0000000020
```

The THL files have the format `thl.data.########`, and the sequence number increases for each new log file. The size of each log file is controlled by the `--thl-log-file-size` [428] configuration parameter. The log files are automatically managed by Tungsten Replicator, with old files automatically removed according to the retention policy set by the `--thl-log-retention` [428] configuration parameter. The files can be manually purged or moved. See Section D.1.5.1, “Purging THL Log Information on a Replica”.

The THL can be viewed and managed by using the thl command. For more information, see Section 8.19, “The thl Command”.

### E.1.1. THL Format

The THL is stored on disk in a specific format that combines the information about the SQL and row data, metadata about the environment in which the row changes and SQL changes were made (metadata), and the log specific information, including the source, database, and timestamp of the information.

A sample of the output is shown below, the information is taken from the output of the thl command:

```
SEQ# = 0 / FRAG# = 0 (last frag)
- TIME = 2013-03-21 18:47:39.0
- EPOCH# = 0
- EVENTID = mysql-bin.000010:0000000000000439;0
- SOURCEID = host1
- METADATA = [mysql_server_id=10;dbms_type=mysql;is_metadata=true;service=dsone;»
    shard=tungsten_firstcluster;heartbeat=MASTER_ONLINE]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
```

```
- OPTIONS = [##charset = ISO8859_1, autocommit = 1, sql_auto_is_null = 0, »
    foreign_key_checks = 1, unique_checks = 1, sql_mode = '', character_set_client = 8, »
    collation_connection = 8, collation_server = 8]
- SCHEMA = tungsten_dsone
- SQL(0) = UPDATE tungsten_dsone.heartbeat SET source_tstamp= '2013-03-21 18:47:39', salt= 1, »
    name= 'MASTER_ONLINE'  WHERE id= 1 /* ___SERVICE___ = [firstcluster] */
```

The sample above shows the information for the SQL executed on a MySQL server. The *EVENTID* [550] shows the MySQL binary log from which the statement has been read. The MySQL server has stored the information in the binary log using `STATEMENT` or `MIXED` mode; log events written in `ROW` mode store the individual row differences. A summary of the THL stored format information, including both hidden values and the information included in the thl command output is provided in Table E.1, "THL Event Format".

## Table E.1. THL Event Format

| Displayed Field | Internal Name | Data type | Size | Description |
|---|---|---|---|---|
| - | `record_length` | Integer | 4 bytes | Length of the full record information, including this field |
| - | `record_type` | Byte | 1 byte | Event record type identifier |
| - | `header_length` | Unsigned int | 4 bytes | Length of the header information |
| *SEQ#* [549] | `seqno` | Unsigned long | 8 bytes | Log sequence number, a sequential value given to each log entry |
| *FRAG#* [549] | `fragno` | Unsigned short | 2 bytes | Event fragment number. An event can consist of multiple fragments of SQL or row log data |
| - | `last_frag` | Byte | 1 byte | Indicates whether the fragment is the last fragment in the sequence |
| *EPOCH#* [550] | `epoch_number` | Unsigned long | 8 bytes | Event epoch number. Used to identify log sections within the Primary THL |
| *SOURCEID* [550] | `source_id` | UTF-8 String | Variable (null terminated) | Event source ID, the hostname or identity of the dataserver that generated the event |
| *EVENTID* [550] | `event_id` | UTF-8 String | Variable (null terminated) | Event ID; in MySQL, for example, the binlog filename and position that contained the original event |
| *SHARDID* [551] | `shard_id` | UTF-8 String | Variable (null terminated) | Shard ID to which the event belongs |
| *TIME* [550] | `tstamp` | Unsigned long | 8 bytes | Time of the commit that triggered the event |
| *FILE* | - | String | - | Filename of the THL file containing the event |
| - | `data_length` | Unsigned int | 4 bytes | Length of the included event data |
| - | `event` | Binary | Variable | Serialized Java object containing the SQL or ROW data |
| *METADATA* [550] | Part of `event` | - | - | Metadata about the event |
| *TYPE* [550] | Part of `event` | - | - | Internal storage type of the event |
| *OPTIONS* [550] | Part of `event` | - | - | Options about the event operation |
| *SCHEMA* [551] | Part of `event` | - | - | Schema used in the event |
| *SQL* [551] | Part of `event` | - | - | SQL statement or row data |
| - | `crc_method` | Byte | 1 byte | Method used to compute the CRC for the event. |
| - | `crc` | Unsigned int | 4 bytes | CRC of the event record (not including the CRC value) |

- *SEQ#* [549] and *FRAG#* [549]

  Individual events within the log are identified by a sequential *SEQUENCE* [549] number. Events are further divided into individual fragments. Fragments are numbered from 0 within a given sequence number. Events are applied to the database wholesale, fragments are used to divide up the size of the statement or row information within the log file. The fragments are stored internally in memory before being applied to the database and therefore memory usage is directly affected by the size and number of fragments held in memory.

  The sequence number as generated during this process is unique and therefore acts as a global transaction ID across a cluster. It can be used to determine whether the Replicas and Primary are in sync, and can be used to identify individual transactions within the replication stream.

- *EPOCH#* [550]

  The *EPOCH* [550] value is used a check to ensure that the logs on the Replica and the Primary match. The *EPOCH* [550] is stored in the THL, and a new *EPOCH* [550] is generated each time a Primary goes online. The *EPOCH* [550] value is then written and stored in the THL alongside each individual event. The *EPOCH* [550] acts as an additional check, beyond the sequence number, to validate the information between the Replica and the Primary. The *EPOCH* [550] value is used to prevent the following situations:

  - In the event of a failover where there are events stored in the Primary log, but which did not make it to a Replica, the *EPOCH* [550] acts as a check so that when the Primary rejoins as the Replica, the *EPOCH* [550] numbers will not match the Replica and the new Primary. The trapped transactions be identified by examining the THL output.
  - When a Replica joins a Primary, the existence of the *EPOCH* [550] prevents the Replica from accepting events that happen to match only the sequence number, but not the corresponding *EPOCH* [550].

  Each time a Tungsten Replicator Primary goes online, the *EPOCH* [550] number is incremented. When the Replica connects, it requests the *SEQUENCE* [549] and *EPOCH* [550], and the Primary confirms that the requested *SEQUENCE* [549] has the requested *EPOCH* [550]. If not, the request is rejected and the Replica gets a validation error:

  ```
  pendingExceptionMessage: Client handshake failure: Client response validation failed: »
      Log epoch numbers do not match: client source ID=west-db2 seqno=408129 »
      server epoch number=408128 client epoch number=189069
  ```

  When this error occurs, the THL should be examined and compared between the Primary and Replica to determine if there really is a mismatch between the two databases. For more information, see Section 7.5, “Managing Transaction Failures”.

- *SOURCEID* [550]

  The *SOURCEID* [550] is a string identifying the source of the event stored in the THL. Typically it is the hostname or host identifier.

- *EVENTID* [550]

  The *EVENTID* [550] is a string identifying the source of the event information in the log. Within a MySQL installed, the *EVENTID* [550] contains the binary log name and position which provided the original statement or row data.

  > **Note**
  >
  > The event ID shown is the end of the corresponding event stored in the THL, not the beginning. When examining the mysqlbinlog for an sequence ID in the THL, you should check the EVENTID of the previous THL sequence number to determine where to start looking within the binary log.

- *TIME* [550]

  When the source information is committed to the database, that information is stored into the corresponding binary log (MySQL) or CDC (Oracle). That information is stored in the THL. The time recorded in the THL is the time the data was committed, *not* the time the data was recorded into the log file.

  The *TIME* [550] value as stored in the THL is used to compute latency information when reading and applying data on a Replica.

- *METADATA* [550]

  Part of the binary *EVENT* payload stored within the event fragment, the metadata is collected and stored in the fragment based on information generated by the replicator. The information is stored as a series of key/value pairs. Examples of the information stored include:

  - MySQL server ID
  - Source database type
  - Name of the Replicator service that generated the THL
  - Any 'heartbeat' operations sent through the replicator service, including those automatically generated by the service, such as when the Primary goes online
  - The name of the shard to which the event belongs
  - Whether the contained data is safe to be applied through a block commit operation

- *TYPE* [550]

  The stored event type. Replicator has the potential to use a number of different stored formats for the THL data. The default type is based on the `com.continuent.tungsten.replicator.event.ReplDBMSEvent`.

- *OPTIONS* [550]

Part of the *EVENT* binary payload, the *OPTIONS* [550] include information about the individual event that have been extracted from the database. These include settings such as the autocommit status, character set and other information, which is used when the information is applied to the database.

There will be one *OPTIONS* [550] block for each *SQL* [551] statement stored in the event.

- *SCHEMA* [551]

  Part of the *EVENT* structure, the *SCHEMA* [551] provides the database or schema name in which the statement or row data was applied.

- *SHARDID* [551]

  When using parallel apply, provides the generated shard ID for the event when it is applied by the parallel applier thread. data.

- *SQL* [551]

  For statement based events, the SQL of the statement that was recorded. Multiple individual SQL statements as part of a transaction can be contained within a single event fragment.

  For example, the MySQL statement:

```
mysql> INSERT INTO user VALUES (null, 'Charles', now());
Query OK, 1 row affected (0.01 sec)
```

  Stores the following into the THL:

```
SEQ# = 3583 / FRAG# = 0 (last frag)
- TIME = 2013-05-27 11:49:45.0
- EPOCH# = 2500
- EVENTID = mysql-bin.000007:0000000625753960;0
- SOURCEID = host1
- METADATA = [mysql_server_id=1687011;dbms_type=mysql;service=firstrep;shard=test]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- SQL(0) = SET INSERT_ID = 3
- OPTIONS = [##charset = ISO8859_1, autocommit = 1, sql_auto_is_null = 0, »
    foreign_key_checks = 1, unique_checks = 1, sql_mode = '', character_set_client = 8, »
    collation_connection = 8, collation_server = 8]
- SCHEMA = test
- SQL(1) = INSERT INTO user VALUES (null, 'Charles', now()) /* ___SERVICE___ = [firstrep] */
```

  For row based events, the information is further defined by the individual row data, including the action type (`UPDATE`, `INSERT` or `DELETE`), *SCHEMA* [551], *TABLE* [551] and individual ROW data. For each ROW, there may be one or more *COL* [551] (column) and identifying *KEY* [551] event to identify the row on which the action is to be performed.

  The same statement when recorded in *ROW* [551] format:

```
SEQ# = 3582 / FRAG# = 0 (last frag)
- TIME = 2013-05-27 11:45:19.0
- EPOCH# = 2500
- EVENTID = mysql-bin.000007:0000000625753710;0
- SOURCEID = host1
- METADATA = [mysql_server_id=1687011;dbms_type=mysql;service=firstrep;shard=test]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- SQL(0) =
 - ACTION = INSERT
 - SCHEMA = test
 - TABLE = user
 - ROW# = 0
  - COL(1: ) = 2
  - COL(2: ) = Charles
  - COL(3: ) = 2013-05-27 11:45:19.0
```

# E.2. Generated Field Reference

When using any of the tools within Tungsten Cluster status information is output using a common set of fields that describe different status information. These field names and terms are constant throughout all of the different tools. A description of each of these different fields is provided below.

## E.2.1. Terminology: Fields *masterConnectUri*

The URI being used to extract THL information. On a Primary, the information may be empty, or may contain the reference to the underlying extractor source where information is being read.

On a Replica, the URI indicates the host from which THL data is being read:

```
masterConnectUri : thl://host1:2112/
```

In a secure installation where SSL is being used to exchange data, the URI protocol will be `thls`:

```
masterConnectUri : thls://host1:2112/
```

## E.2.2. Terminology: Fields *masterListenUri*

The URI on which the replicator is listening for incoming Replica requests. On a Primary, this is the URI used to distribute THL information.

```
masterListenUri : thls://host1:2112/
```

## E.2.3. Terminology: Fields *accessFailures*

## E.2.4. Terminology: Fields *active*

## E.2.5. Terminology: Fields *activeSeqno*

## E.2.6. Terminology: Fields *appliedLastEventId*

The event ID from the source database of the last corresponding event from the stage that has been applied to the database.

MySQL

When extracting from MySQL, the output from trepctl shows the MySQL binary log file and the byte position within the log where the transaction was extracted:

```
shell> trepctl status
Processing status command...
NAME VALUE
---- -----
appliedLastEventId : mysql-bin.000064:0000000002757461;0
...
```

Oracle CDC

When extracting from Oracle using the CDC method, the event ID is composed of the Oracle SCN number:

```
NAME VALUE
---- -----
appliedLastEventId : ora:16626156
```

Oracle Redo Reader

When extracting from Oracle using the Redo Reader method, the event ID is composed of a combination of Oracle SCN, transaction, and PLOG file numbers, separated by a hash symbol:

```
NAME VALUE
---- -----
appliedLastEventId : 8931871791244#0018.002.000196e1#LAST#8931871791237#100644
```

The format is:

```
COMMITSCN#XID#LCR#MINSCN#PLOGSEQ
```

- COMMITSCN

  Last committed Oracle System Change Number (SCN).

- XID

  Transaction ID.

- LCR

  Last committed record number.

- MINSCN

  Minimum stored Oracle SCN.

- PLOGSEQ

  PLOG file sequence number.

## E.2.7. Terminology: Fields *appliedLastSeqno*

The last sequence number for the transaction from the Tungsten stage that has been applied to the database. This indicates the last actual transaction information written into the Replica database.

```
appliedLastSeqno : 212
```

When using parallel replication, this parameter returns the minimum applied sequence number among all the channels applying data.

## E.2.8. Terminology: Fields *appliedLatency*

The *appliedLatency* is the latency between the commit time of the source event and the time the last committed transaction reached the end of the corresponding pipeline within the replicator.

Within a Primary, this indicates the latency between the transaction commit time and when it was written to the THL. In a Replica, it indicates the latency between the commit time on the Primary database and when the transaction has been committed to the destination database. Clocks must be synchronized across hosts for this information to be accurate.

```
appliedLatency : 0.828
```

The latency is measure in seconds. Increasing latency may indicate that the destination database is unable to keep up with the transactions from the Primary.

In replicators that are operating with parallel apply, *appliedLatency* indicates the latency of the trailing channel. Because the parallel apply mechanism does not update all channels simultaneously, the figure shown may trail significantly from the actual latency.

## E.2.9. Terminology: Fields *applier.class*

Classname of the current applier engine

## E.2.10. Terminology: Fields *applier.name*

Name of the current applier engine

## E.2.11. Terminology: Fields *applyTime*

## E.2.12. Terminology: Fields *autoRecoveryEnabled*

Indicates whether autorecovery has been enabled by setting the `--auto-recovery-max-attempts` [395]. The field indicates the value as either `true` or `false` accordingly.

## E.2.13. Terminology: Fields *autoRecoveryTotal*

A count of the number of times the replicator has used autorecovery to go back online since the replicator was started. This can be used to determine if autorecovery has been used. More details on autorecovery can be found in the `trepsvc.log` file.

The counter is reset when the replicator determines that the replicator has successfully gone online after an autorecovery.

## E.2.14. Terminology: Fields *averageBlockSize*

## E.2.15. Terminology: Fields *blockCommitRowCount*

## E.2.16. Terminology: Fields *cancelled*

## E.2.17. Terminology: Fields *channel*

## E.2.18. Terminology: Fields *channels*

The number of channels being used to apply transactions to the target dataserver. In a standard replication setup there is typically only one channel. When parallel replication is in effect, there will be more than one channel used to apply transactions.

```
channels : 1
```

## E.2.19. Terminology: Fields *clusterName*

The name of the cluster. This information is different to the service name and is used to identify the cluster, rather than the individual service information being output.

## E.2.20. Terminology: Fields *commits*

## E.2.21. Terminology: Fields *committedMinSeqno*

## E.2.22. Terminology: Fields *criticalPartition*

## E.2.23. Terminology: Fields *currentBlockSize*

## E.2.24. Terminology: Fields *currentEventId*

Event ID of the transaction currently being processed

## E.2.25. Terminology: Fields *currentLastEventId*

## E.2.26. Terminology: Fields *currentLastFragno*

## E.2.27. Terminology: Fields *currentLastSeqno*

## E.2.28. Terminology: Fields *currentTimeMillis*

The current time on the host, in milliseconds since the epoch. This information can used to confirm that the time on different hosts is within a suitable limit. Internally, the information is used to record the time when transactions are applied, and may therefore the *appliedLatency* figure.

## E.2.29. Terminology: Fields *dataServerHost*

## E.2.30. Terminology: Fields *discardCount*

## E.2.31. Terminology: Fields *doChecksum*

## E.2.32. Terminology: Fields *estimatedOfflineInterval*

## E.2.33. Terminology: Fields *eventCount*

## E.2.34. Terminology: Fields *extensions*

## E.2.35. Terminology: Fields *extractTime*

## E.2.36. Terminology: Fields *extractor.class*

## E.2.37. Terminology: Fields *extractor.name*

## E.2.38. Terminology: Fields *filter.#.class*

## E.2.39. Terminology: Fields *filter.#.name*

## E.2.40. Terminology: Fields *filterTime*

## E.2.41. Terminology: Fields *flushIntervalMillis*

## E.2.42. Terminology: Fields *fsyncOnFlush*

## E.2.43. Terminology: Fields *headSeqno*

## E.2.44. Terminology: Fields *intervalGuard*

## E.2.45. Terminology: Fields *lastCommittedBlockSize*

The *lastCommittedBlockSize* contains the size of the last block that was committed as part of the block commit procedure. The value is only displayed on appliers and defines the number of events in the last block. By comparing this value to the configured block commit size, the commit type can be determined.

For more information, see Section 12.1, “Block Commit”.

## E.2.46. Terminology: Fields *lastCommittedBlockTime*

The *lastCommittedBlockSize* contains the duration since the last committed block. The value is only displayed on appliers and defines the number of seconds since the last block was committed. By comparing this value to the configured block interval, the commit type can be determined.

For more information, see Section 12.1, “Block Commit”.

## E.2.47. Terminology: Fields *latestEpochNumber*

## E.2.48. Terminology: Fields *logConnectionTimeout*

## E.2.49. Terminology: Fields *logDir*

## E.2.50. Terminology: Fields *logFileRetainMillis*

## E.2.51. Terminology: Fields *logFileSize*

## E.2.52. Terminology: Fields *maxChannel*

## E.2.53. Terminology: Fields *maxDelayInterval*

## E.2.54. Terminology: Fields *maxOfflineInterval*

## E.2.55. Terminology: Fields *maxSize*

## E.2.56. Terminology: Fields *maximumStoredSeqNo*

The maximum transaction ID that has been stored locally on the machine in the THL. Because Tungsten Replicator operates in stages, it is sometimes important to compare the sequence and latency between information being ready from the source into the THL, and then from the THL into the database. You can compare this value to the *appliedLastSeqno*, which indicates the last sequence committed to the database. The information is provided at a resolution of milliseconds.

```
maximumStoredSeqNo : 25
```

## E.2.57. Terminology: Fields *minimumStoredSeqNo*

The minimum transaction ID stored locally in the THL on the host:

```
minimumStoredSeqNo : 0
```

The figure should match the lowest transaction ID as output by the thl index command. On a busy host, or one where the THL information has been purged, the figure will show the corresponding transaction ID as stored in the THL.

## E.2.58. Terminology: Fields *name*

## E.2.59. Terminology: Fields *offlineRequests*

Contains the specifications of one or more future offline events that have been configured for the replicator. Multiple events are separated by a semicolon:

```
shell> trepctl status
...
inimumStoredSeqNo : 0
offlineRequests : Offline at sequence number: 5262;Offline at time: 2014-01-01 00:00:00 EST
pendingError : NONE
```

## E.2.60. Terminology: Fields *otherTime*

## E.2.61. Terminology: Fields *pendingError*

## E.2.62. Terminology: Fields *pendingErrorCode*

## E.2.63. Terminology: Fields *pendingErrorEventId*

## E.2.64. Terminology: Fields *pendingErrorSeqno*

The sequence number where the current error was identified

## E.2.65. Terminology: Fields *pendingExceptionMessage*

The current error message that caused the current replicator offline

## E.2.66. Terminology: Fields *pipelineSource*

The source for data for the current pipeline. On a Primary, the pipeline source is the database that the Primary is connected to and extracting data from. Within a Replica, the pipeline source is the Primary replicator that is providing THL data.

## E.2.67. Terminology: Fields *processedMinSeqno*

## E.2.68. Terminology: Fields *queues*

## E.2.69. Terminology: Fields *readOnly*

## E.2.70. Terminology: Fields *relativeLatency*

The relativeLatency is the latency between now and timestamp of the last event written into the local THL. This information gives an indication of how fresh the incoming THL information is. On a Primary, it indicates whether the Primary is keeping up with transactions generated on the Primary database. On a Replica, it indicates how up to date the THL read from the Primary is.

A large value can either indicate that the database is not busy, that a large transaction is currently being read from the source database, or from the Primary replicator, or that the replicator has stalled for some reason.

An increasing `relativeLatency` on the Replica may indicate that the replicator may have stalled and stopped applying changes to the dataserver.

## E.2.71. Terminology: Fields *resourcePrecedence*

## E.2.72. Terminology: Fields *rmiPort*

## E.2.73. Terminology: Fields *role*

The current role of the host in the corresponding service specification. Primary roles are `master` and `slave`.

## E.2.74. Terminology: Fields *seqnoType*

The internal class used to store the transaction ID. In MySQL replication, the sequence number is typically stored internally as a Java Long (`java.lang.Long`). In heterogeneous replication environments, the type used may be different to match the required information from the source database.

## E.2.75. Terminology: Fields *serializationCount*

## E.2.76. Terminology: Fields *serialized*

## E.2.77. Terminology: Fields *serviceName*

The name of the configured service, as defined when the deployment was first created through tpm.

```
serviceName : alpha
```

A replicator may support multiple services. The information is output to confirm the service information being displayed.

## E.2.78. Terminology: Fields *serviceType*

The configured service type. Where the replicator is on the same host as the database, the service is considered to be `local`. When reading or write to a remote dataserver, the service is `remote`.

## E.2.79. Terminology: Fields *shard_id*

## E.2.80. Terminology: Fields *simpleServiceName*

A simplified version of the *serviceName*.

## E.2.81. Terminology: Fields *siteName*

## E.2.82. Terminology: Fields *sourceId*

## E.2.83. Terminology: Fields *stage*

## E.2.84. Terminology: Fields *started*

## E.2.85. Terminology: Fields *state*

## E.2.86. Terminology: Fields *stopRequested*

## E.2.87. Terminology: Fields *store.#*

## E.2.88. Terminology: Fields *storeClass*

## E.2.89. Terminology: Fields *syncInterval*

## E.2.90. Terminology: Fields *taskCount*

## E.2.91. Terminology: Fields *taskId*

## E.2.92. Terminology: Fields *timeInCurrentEvent*

Shows the time that the replicator has been processing the current event. When processing very large transactions this can be used to determine whether the replicator has stalled or is still actively extracting or applying the information.

## E.2.93. Terminology: Fields *timeInStateSeconds*

## E.2.94. Terminology: Fields *timeoutMillis*

## E.2.95. Terminology: Fields *totalAssignments*

## E.2.96. Terminology: Fields *transitioningTo*

## E.2.97. Terminology: Fields *uptimeSeconds*

## E.2.98. Terminology: Fields *version*

# Appendix F. Internals

Tungsten Cluster includes a number of different systems and elements to provide the core services and functionality. Some of these are designed only to be customer-configured. Others should be changed only on the advice of Continuent or Continuent support. This chapter covers a range of different systems hat are designated as internal features and functionality.

This chapter contains information on the following sections of Tungsten Cluster:

- Section F.1, "Extending Backup and Restore Behavior" — details on how the backup scripts operate and how to write custom backup scripts.
- Section F.2, "Character Sets in Database and Tungsten Cluster" — covers how character sets affect replication and command-line tool output.
- Section F.4, "Memory Tuning and Performance" — information on how the memory is used and allocated within Tungsten Cluster.

## F.1. Extending Backup and Restore Behavior

The backup and restore system within Tungsten Cluster is handled entirely by the replicator. When a backup is initiated, the replicator on the specified datasource is asked to start the backup process.

The backup and restore system both use a modular mechanism that is used to perform the actual backup or restore operation. This can be configured to use specific backup tools or a custom script.

### F.1.1. Backup Behavior

When a backup is requested, the Tungsten Replicator performs a number of separate, discrete, operations designed to perform the backup operation.

The backup operation performs the following steps:

1. Tungsten Replicator identifies the filename where properties about the backup will be stored. The file is used as the primary interface between the underlying backup script and Tungsten Replicator.
2. Tungsten Replicator executes the configured backup/restore script, supplying any configured arguments, and the location of a properties file, which the script updates with the location of the backup file created during the process.
3. If the backup completes successfully, the file generated by the backup process is copied into the configured Tungsten Cluster directory (for example `/opt/continuent/backups`.
4. Tungsten Replicator updates the property information with a CRC value for the backup file and the standard metadata for backups, including the tool used to create the backup.

A log is created of the backup process into a file according to the configured backup configuration. For example, when backing up using mysqldump the log is written to the log directory as `mysqldump.log`. When using a custom script, the log is written to `script.log`.

As standard, Tungsten Replicator supports two primary backup types, mysqldump and xtrabackup. A third option is based on the incremental version of the xtrabackup tool. The use of external backup script enables additional backup tools and methods to be supported.

To create a custom backup script, see Section F.1.3, "Writing a Custom Backup/Restore Script" for a list of requirements and samples.

### F.1.2. Restore Behavior

The restore operation operates in a similar manner to the backup operation. The same script is called (but supplied with the `-restore` command-line option).

The restore operation performs the following steps:

1. Tungsten Replicator creates a temporary properties file, which contains the location of the backup file to be restored.
2. Tungsten Replicator executes the configured backup/restore script in restore mode, supplying any configured arguments, and the location of the properties file.
3. The script used during the restore process should read the supplied properties file to determine the location of the backup file.
4. The script performs all the necessary steps to achieve the restore process, including stopping the dataserver, restoring the data, and restarting the dataserver.

5. The replicator will remain in the `OFFLINE` [195] state once the restore process has finished.

## F.1.3. Writing a Custom Backup/Restore Script

The synopsis of the custom script is as follows:

`SCRIPT` {`-backup-restore`} -properties `FILE` -options `OPTIONS`

Where:

- `-backup` — indicates that the script should work in the backup mode and create a backup.
- `-restore` — indicates that the scrip should work in the restore mode and restore a previous backup.
- `-properties` — defines the name of the properties file. When called in *backup* mode, the properties file should be updated by the script with the location of the generated backup file. When called in *restore* mode, the file should be examined by the script to determine the backup file that will be used to perform the restore operation.
- `-options` — specifies any unique options to the script.

The custom script must support the following:

- The script must be capable of performing both the backup and the restore operation. Tungsten Replicator selects the operation by providing the `-backup` or `-restore` option to the script on the command-line.
- The script must parse command-line arguments to extract the operation type, properties file and other settings.
- Accept the name of the properties file to be used during the backup process. This is supplied on the command-line using the format:

  ```
  -properties FILENAME
  ```

  The properties file is used by Tungsten Replicator to exchange information about the backup or restore.
- Must parse any additional options supplied on the command-line using the format:

  ```
  -options ARG1=VAL1&ARG2=VAL2
  ```
- Must be responsible for executing whatever steps are required to create a consistent snapshot of the dataserver
- Must place the contents of the database backup into a single file. If the backup process generates multiple files, then the contents should be packaged using tar or zip.

  The script has to determine the files that were generated during the backup process and collect them into a single file as appropriate.
- Must update the supplied properties with the name of the backup file generated, as follows:

  ```
  file=BACKUPFILE
  ```

  If the file has not been updated with the information, or the file cannot be found, then the backup is considered to have failed.

  Once the backup process has completed, the backup file specified in the properties file will be moved to the configured backup location (for example `/opt/continuent/backups`).
- Tungsten Replicator will forward all `STDOUT` and `STDERR` from the script to the log file `script.log` within the log directory. This file is recreated each time a backup is executed.
- Script should have an exit (return) value of 0 for success, and 1 for failure. The script is responsible for handling any errors in the underlying backup tool or script used to perform the backup, but it must then pass the corresponding success or failure condition using the exit code.

A sample Ruby script that creates a simple text file as the backup content, but demonstrates the core operations for the script is shown below:

```
#!/usr/bin/env ruby
require "/opt/continuent/tungsten/cluster-home/lib/ruby/tungsten"
require "/opt/continuent/tungsten/tungsten-replicator/lib/ruby/backup"
class MyCustomBackupScript < TungstenBackupScript
  def backup
    TU.info("Take a backup with arg1 = #{@options[:arg1]} and myarg = #
{@options[:myarg]}")
    storage_file = "/opt/continuent/backups/backup_" +
Time.now.strftime("%Y-%m-%d_%H-%M") + "_" + rand(100).to_s()
    # Take a backup of the server and store the information to
storage_file
    TU.cmd_result("echo 'my backup' > #{storage_file}")
```

```
    # Write the filename to the final storage file
    TU.cmd_result("echo \"file=#{storage_file}\" > #
{@options[:properties]}")
end
  def restore
    storage_file = TU.cmd_result(". #{@options[:properties]}; echo
$file")
    TU.info("Restore a backup from #{storage_file} with arg1 = #
{@options[:arg1]} and myarg = #{@options[:myarg]}")
    # Process the contents of storage_file to restore into the database
server
end
```

An alternative script using Perl is provided below:

```
#!/usr/bin/perl

use strict;
use warnings;
use Getopt::Long;
use IO::File;

my $argstring = join(' ',@ARGV);

my ($backup,$restore,$properties,$options) = (0,0,'','');

my $result = GetOptions("backup" => \$backup,
   "restore" => \$restore,
   "properties=s" => \$properties,
   "options=s" => \$options,
    );

if ($backup)
{
    my ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) = localtime(time);
    my $backupfile = sprintf('mcbackup.%04d%02d%02d-%02d%02d%02d-%02d.dump',
        ($year+1900),$mon,$mday,$hour,$min,$sec,$$);

    my $out = IO::File->new($backupfile,'w') or die "Couldn't open the backup file: $backupfile";

# Fake backup data

    print $out "Backup data!\n";

    $out->close();

# Update the properties file
    my $propfile = IO::File->new($properties,'w') or die "Couldn't write to the properties file";
    print $propfile "file=$backupfile\n";
    $propfile->close();
}

if ($restore)
{
    warn "Would be restoring information using $argstring\n";
}

exit 0;
```

## F.1.4. Enabling a Custom Backup Script

To enable a custom backup script, the installation must be updated through tpm to use the script backup method. To update the configuration:

1. Create or copy the backup script into a suitable location, for example `/opt/continuent/share`.
2. Copy the script to each of the datasources within your dataservice.
3. Update the configuration using tpm. The `--repl-backup-method` [396] should be set to *`script`*, and the directory location set using the `--repl-backup-script` [397] option:

   ```
   shell> ./tools/tpm update --repl-backup-method=script \
       --repl-backup-script=/opt/continuent/share/mcbackup.pl \
       --repl-backup-online=true
   ```

   The `--repl-backup-online` [397] option indicates whether the backup script operates in online or offline mode. If set to false, replicator must be in the offline state because the backup process is started.

   To pass additional arguments or options to the script, use the *`replicator.backup.agent.script.options`* property to supply a list of ampersand separate key/value pairs, for example:

```
--property=replicator.backup.agent.script.options="arg1=val1&myarg=val2"
```

These are the custom parameters which are supplied to the script as the value of the `-options` parameter when the script is called.

Once the backup script has been enabled within the configuration it can be used when performing a backup through the standard backup or restore interface:

```
shell> trepctl -host host2 backup -backup script
```

### Note

Note that the name of the backup method is `script`, not the actual name of the script being used.

# F.2. Character Sets in Database and Tungsten Cluster

Character sets within the databases and within the configuration for Java and the wrappers for Tungsten Cluster must match to enable the information to be extracted and viewed.

For example, if you are extracting with the UTF-8 character set, the data must be applied to the target database using the same character set. In addition, the Tungsten Replicator should be configured with a corresponding matching character set. For installations where replication is between identical database flavours (for example, MySQL and MySQL) no explicit setting should be made. For heterogeneous deployments, the character set should be set explicitly.

When installing and using Tungsten Cluster, be aware of the following aspects when using character sets:

- When installing Tungsten Cluster, use the `--java-file-encoding` [410] to tpm to configure the character set.
- When using the thl command, the character set may need to be explicitly stated to view the content correctly:

```
shell> thl list -charset utf8
```

For more information on setting character sets within your database, see your documentation for the database:

- MySQL

For more information on the character set names and support within Java, see:

- Java 8 SE

# F.3. Understanding Replication of Date/Time Values

- Replicator processes default to UTC internally by setting the Java VM default time zone to UTC. This default can be changed by setting the replicator.time_zone property in the replicator services.propertiesx file but is not recommended other than for problem diagnosis or specialized testing.
- Replicas store a time zone on statements and row changes extracted from MySQL.
- Replicators use UTC as the session time zone when applying to MySQL replicas.
- Replicators similarly default to UTC when applying transactions to data warehouses like Hadoop, Vertica, or Amazon Redshift.
- The thl utility prints time-related data using the default GMT time zone. This can be altered using the `-timezone` option.

## Best Practices

We recommend the following steps to ensure successful replication of time-related data.

- Standardize all DBMS server and host time zones to UTC. This minimizes time zone inconsistencies between applications and data stores. The recommendation is particularly important when replicating between different DBMS types, such as MySQL to Hadoop.
- Use the default time zone settings for Tungsten replicator. Do not change the time zones unless specifically recommended by Continuent support.
- If you cannot standardize on UTC at least ensure that time zones are set consistently on all hosts and applications.

Arbitrary time zone settings create a number of corner cases for database management beyond replication. Standardizing on UTC helps minimize them, hence is strongly recommended.

# F.4. Memory Tuning and Performance

Different areas of Tungsten Cluster use memory in different ways, according to the operation and requirements of the component. Specific information on how memory is used by different components and how it is used is available below:

- Tungsten Replicator — Memory performance and tuning options.

## F.4.1. Understanding Tungsten Replicator Memory Tuning

Replicators are implemented as Java processes, which use two types of memory: stack space, which is allocated per running thread and holds objects that are allocated within individual execution stack frames, and heap memory, which is where objects that persist across individual method calls live. Stack space is rarely a problem for Tungsten as replicators rarely run more than 200 threads and use limited recursion. The Java defaults are almost always sufficient. Heap memory on the other hand runs out if the replicator has too many transactions in memory at once. This results in the dreaded Java OutOfMemory exception, which causes the replicator to stop operating. When this happens you need to look at tuning the replicator memory size.

To understand replicator memory usage, we need to look into how replicators work internally. Replicators use a "pipeline" model of execution that streams transactions through 1 or more concurrently executing stages. As you can can see from the attached diagram, a Replica pipeline might have a stage to read transactions to the Primary and put them in the THL, a stage to read them back out of the THL into an in-memory queue, and a stage to apply those transactions to the Replica. This model ensures high performance as the stages work independently. This streaming model is quite efficient and normally permits Tungsten to transfer even exceedingly large transactions, as the replicator breaks them up into smaller pieces called transaction fragments.

The pipeline model has consequences for memory management. First of all, replicators are doing many things at one, hence need enough memory to hold all current objects. Second, the replicator works fastest if the in-memory queues between stages are large enough that they do not ever become empty. This keeps delays in upstream processing from delaying things at the end of the pipeline. Also, it allows replicators to make use of block commit. Block commit is an important performance optimization in which stages try to commit many transactions at once on Replicas to amortize the cost of commit. In block commit the end stage continues to commit transactions until it either runs out of work (i.e., the upstream queue becomes empty) or it hits the block commit limit. Larger upstream queues help keep the end stage from running out of work, hence increase efficiency.

Bearing this in mind, we can alter replicator behavior in a number of ways to make it use less memory or to handle larger amounts of traffic without getting a Java OutOfMemory error. You should look at each of these when tuning memory:

- Property wrapper.java.memory in file `wrapper.conf`. This controls the amount of heap memory available to replicators. 1024 MB is the minimum setting for most replicators. Busy replicators, those that have multiple services, or replicators that use parallel apply should consider using 2048 MB instead. If you get a Java OutOfMemory exception, you should first try raising the current setting to a higher value. This is usually enough to get past most memory-related problems. You can set this at installation time as the `--repl-java-mem-size` [410] parameter.

  If you set the heap memory to a very large value (e.g. over 3 GB), you should also consider enabling concurrent garbage collection. Java by default uses mark-and-sweep garbage collection, which may result in long pauses during which network calls to the replicator may fail. Concurrent garbage collection uses more CPU cycles and reduces on-going performance a bit but avoids periods of time during which the replicator is non-responsive. You can set this using the `--repl-java-enable-concurrent-gc` [409] parameter at installation time.)

- Property *`replicator.global.buffer.size`*.. This controls two things, the size of in-memory queues in the replicator as well as the block commit size. If you still have problems after increasing the heap size, try reducing this value. It reduces the number of objects simultaneously stored on the Java heap. A value of 2 is a good setting to try to get around temporary problems. This can be set at installation time as the `--repl-buffer-size` [398] parameter.

- Property replicator.stage.q-to-dbms.blockCommitRowCount in the replicator properties file. This parameter sets the block commit count in the final stage in a Replica pipeline. If you reduce the global buffer size, it is a good idea to set this to a fixed size, such as 10, to avoid reducing the block commit effect too much. Very low block commit values in this stage can cut update rates on Replicas by 50% or more in some cases. This is available at installation time as the `--repl-svc-applier-buffer-size` parameter.

- Property replicator.extractor.dbms.transaction_frag_size in the `replicator.properties` file. This parameter controls the size of fragments for long transactions. Tungsten automatically breaks up long transactions into fragments. This parameter controls the number of bytes of binlog per transaction fragment. You can try making this value smaller to reduce overall memory usage if many transactions are simultaneously present. Normally however this value has minimal impact.

Finally, it is worth mentioning that the main cause of out-of-memory conditions in replicators is large transactions. In particular, Tungsten cannot fragment individual statements or row changes, so changes to very large column values can also result in OutOfMemory conditions. For now the best approach is to raise memory, as described above, and change your application to avoid such transactions.

# F.5. Tungsten Replicator Pipelines and Stages

A pipeline (or service) acts upon data.

Pipelines consist of a variable number of stages.

Every stage's workflow consists of three (3) actions, which are:

- Extract: the source for extraction could be the mysql server binary logs on a Primary, and the local THL on disk for a Replica
- Filter: any configured filters are applied here
- Apply: the apply target can be THL on disk on a Primary, and the database server on a Replica

Stages can be customized with filters, and filters are invoked on a per-stage basis.

By default, there are two pipeline services defined:

- Primary replication service, which contains two (2) stages:
  - binlog-to-q: reads information from the MySQL binary log and stores the information within an in-memory queue.
  - q-to-thl: in-memory queue is written out to the THL file on disk.
- Replica replication service, which contains three (3) stages:
  - remote-to-thl: remote THL information is read from a Primary datasource and written to a local file on disk.
  - thl-to-q: THL information is read from the file on disk and stored in an in-memory queue.
  - q-to-dbms: data from the in-memory queue is written to the target database.

## F.6. Tungsten Cluster Schemas

# Appendix G. Frequently Asked Questions (FAQ)

G.1. On a Tungsten Replicator Replica, how do I set both the local Replica THL listener port and the upstream Primaries THL listener port?

You need to specify two options: `thl-port` [429] to set the Replica THL listener port and `master-thl-port` [413] to define the upstream Primary THL listener port. Otherwise `thl-port` [429] alone sets BOTH.

G.2. One of my hosts is regularly a number of seconds behind my other Replicas?

The most likely culprit for this issue is that the time is different on the machine in question. If you have ntp or a similar network time tool installed on your machine, use it to update the current time across *all* the hosts within your deployment:

```
shell> ntpdate pool.ntp.org
```

Once the command has been executed across all the hosts, trying sending a heartbeat on the Primary to Replicas and checking the latency:

```
shell> trepctl heartbeat
```

G.3. Does the replicate filter (i.e. replicate.do and replicate.ignore) address both DML and DDL?

Both filters replicate.do and replicate.ignore will either do or ignore both DML and DDL

DDL is currently ONLY replicated for MySQL to MySQL or Oracle to Oracle topologies, or within MySQL Clusters, although it would be advisable not to use ignore/do filters in a clustered environment where data/structural integrity is key.

With replicate.do, all DML and DDL will be replicated ONLY for any database or table listed as part of the do filter.

With replicate.ignore, all DML and DDL will be replicated except for any database or table listed as part of the ignore filter.

G.4. How do you change the replicator heap size after installation?

You can change the configuration by running the following command from the staging directory:

```
shell> ./tools/tpm --host=host1 --java-mem-size=2048
```

# Appendix H. Ecosystem Support

In addition to the core utilities provided by Tungsten Cluster, additional tools and scripts are available that augment the core code with additional functionality, such as integrating with third-party monitoring systems, or providing additional functionality that is designed to be used and adapted for specific needs and requirements.

Different documentation and information exists for the following tools:

- Github — a selection of tools and utilities are provided in Github to further support and expand the functionality of Tungsten Cluster during deployment, monitoring, and management.

## H.1. Continuent Github Repositories

In addition to the core product releases, Continuent also support a number of repositories within the Github system.

To access these repositories and use the tools and information within them, use the git command (available from git-scm.com). To copy the repository to a machine, use the clone command, specifying the repository URL:

# Appendix I. Configuration Property Reference