Tungsten Replicator 5.4 Manual

Continuent Ltd
Abstract

This manual documents Tungsten Replicator 5.4. This includes information for:

- Tungsten Replicator
- Tungsten Replicator for Analytics and Big Data

Build date: 2020-02-26 (2c0bfc6b)

Up to date builds of this document: Tungsten Replicator 5.4 Manual [Online], Tungsten Replicator 5.4 Manual [PDF]
Table of Contents

Preface ................................................................. xvi
1. Legal Notice ....................................................... xvi
2. Conventions ......................................................... xvii
3. Quickstart Guide .................................................... xvi
1. Introduction ......................................................... 18
   1.1. Tungsten Replicator ............................................. 19
       1.1.1. Extractor .................................................. 19
       1.1.2. Appliers ................................................ 20
       1.1.3. Transaction History Log [THL] .......................... 20
       1.1.4. Filtering ................................................. 20
2. Deployment Overview ............................................... 22
   2.1. Deployment Sources ............................................. 22
       2.1.1. Using the TAR/GZipped files .............................. 23
       2.1.2. Using the RPM and DEB package files .................... 23
   2.2. Best Practices ................................................. 24
       2.2.1. Best Practices: Deployment .............................. 24
       2.2.2. Best Practices: Operations .............................. 25
       2.2.3. Best Practices: Maintenance ............................ 25
   2.3. Prepare Hosts ................................................. 25
       2.3.1. Prepare MySQL Hosts .................................... 25
       2.3.2. Deploy SSH Keys ....................................... 25
   2.4. Common tpm Options During Deployment ....................... 26
   2.5. Starting and Stopping Tungsten Replicator ..................... 27
   2.6. Configuring Startup on Boot ................................ 27
   2.7. Removing Datasources from a Deployment ..................... 27
       2.7.1. Removing a Datasource from an Existing Deployment .... 28
   2.8. Understanding Deployment Styles and Topologies .............. 28
       2.8.1. Tungsten Replicator Extraction Operation ............... 28
       2.8.2. Understanding Deployment Models ......................... 29
       2.8.3. Understanding Deployment Topologies ..................... 30
           2.8.3.1. Simple Master/Slave Topology .................... 30
           2.8.3.2. Multi-Master Topology ........................... 31
           2.8.3.3. Fan-Out Topology ................................ 32
           2.8.3.4. Fan-In Topology ................................ 32
           2.8.3.5. Replicating in/out of an existing Tungsten Cluster .... 32
       2.9. Understanding Heterogeneous Deployments .................. 33
           2.9.1. How Heterogeneous Replication Works ................. 33
               2.9.1.1. JDBC Applier based Replication ............ 34
               2.9.1.2. Native Applier Replication [e.g. MongoDB, Elasticsearch] ... 35
               2.9.1.3. Batch Loading .................................. 35
               2.9.1.4. Schema Creation and Replication ............. 35
   3. Deploying MySQL Extractors .................................... 36
   3.1. MySQL Replication Pre-Requisites ............................ 36
   3.2. Deploying a Master/Slave Topology ........................... 36
       3.2.1. Monitoring the MySQL Extractor ......................... 43
   3.3. Deploying an Extractor for Amazon RDS/Amazon Aurora ......... 44
       3.3.1. Changing Amazon RDS/Aurora Instance Configurations .......... 49
           3.3.1.1. Changing Amazon RDS using command line functions .. 49
           3.3.1.2. Changing Amazon Aurora Parameters using AWS Console ... 50
       3.4. Replicating Data Out of a Cluster .......................... 55
           3.4.1. Prepare: Replicating Data Out of a Cluster .......... 56
           3.4.2. Deploy: Replicating Data Out of a Cluster ........... 56
   4. Deploying Oracle Extractors ..................................... 60
   4.1. Deploying Oracle Replication using CDC ........................ 60
       4.1.1. How Oracle Extraction Works ........................... 62
       4.1.2. Data Type Differences and Limitations .................. 64
       4.1.3. Creating an Oracle to MySQL Deployment ................. 66
           4.1.3.1. Configuring the Oracle Environment ................ 66
           4.1.3.2. Creating the MySQL Environment ................... 68
           4.1.3.3. Creating the Destination Schema ................... 68
           4.1.3.4. Creating the Master Replicator .................... 68
           4.1.3.5. Creating the Slave Replicator ..................... 71
       4.1.4. Creating an Oracle to Oracle Deployment ................. 73
           4.1.4.1. Setting up the Source Oracle Environment .......... 73

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.9. Deploying the Vertica Applier</td>
<td>155</td>
</tr>
<tr>
<td>5.9.1. Preparing for Vertica Deployments</td>
<td>156</td>
</tr>
<tr>
<td>5.9.2. Install Vertica Applier</td>
<td>157</td>
</tr>
<tr>
<td>5.9.3. Management and Monitoring of Vertica Deployments</td>
<td>161</td>
</tr>
<tr>
<td>5.9.4. Troubleshooting Vertica Installations</td>
<td>162</td>
</tr>
<tr>
<td>6. Advanced Deployments</td>
<td>164</td>
</tr>
<tr>
<td>6.1. Deploying a Fan-In Topology</td>
<td>164</td>
</tr>
<tr>
<td>6.1.1. Management and Monitoring Fan-in Deployments</td>
<td>166</td>
</tr>
<tr>
<td>6.2. Deploying Multiple Replicators on a Single Host</td>
<td>167</td>
</tr>
<tr>
<td>6.2.1. Preparing Multiple Replicators</td>
<td>167</td>
</tr>
<tr>
<td>6.2.2. Install Multiple Replicators</td>
<td>168</td>
</tr>
<tr>
<td>6.2.3. Best Practices: Multiple Replicators</td>
<td>174</td>
</tr>
<tr>
<td>6.3. Replicating Data Into an Existing Dataservice</td>
<td>174</td>
</tr>
<tr>
<td>6.4. Deploying Parallel Replication</td>
<td>177</td>
</tr>
<tr>
<td>6.4.1. Application Prerequisites for Parallel Replication</td>
<td>177</td>
</tr>
<tr>
<td>6.4.2. Enabling Parallel Apply During Install</td>
<td>178</td>
</tr>
<tr>
<td>6.4.3. Channels</td>
<td>180</td>
</tr>
<tr>
<td>6.4.4. Parallel Replication and Offline Operation</td>
<td>181</td>
</tr>
<tr>
<td>6.4.4.1. Clean Offline Operation</td>
<td>181</td>
</tr>
<tr>
<td>6.4.4.2. Tuning the Time to Go Offline Cleanly</td>
<td>181</td>
</tr>
<tr>
<td>6.4.4.3. Unclean Offline</td>
<td>182</td>
</tr>
<tr>
<td>6.4.5. Adjusting Parallel Replication After Installation</td>
<td>182</td>
</tr>
<tr>
<td>6.4.5.1. How to Enable Parallel Apply After Installation</td>
<td>182</td>
</tr>
<tr>
<td>6.4.5.2. How to Change Channels Safely</td>
<td>184</td>
</tr>
<tr>
<td>6.4.5.3. How to Disable Parallel Replication Safely</td>
<td>185</td>
</tr>
<tr>
<td>6.4.5.4. How to Switch Parallel Queue Types Safely</td>
<td>186</td>
</tr>
<tr>
<td>6.4.6. Monitoring Parallel Replication</td>
<td>186</td>
</tr>
<tr>
<td>6.4.6.1. Useful Commands for Parallel Monitoring Replication</td>
<td>186</td>
</tr>
<tr>
<td>6.4.6.2. Parallel Replication and Applied Latency On Slaves</td>
<td>187</td>
</tr>
<tr>
<td>6.4.6.3. Relative Latency</td>
<td>188</td>
</tr>
<tr>
<td>6.4.6.4. Serialization Count</td>
<td>188</td>
</tr>
<tr>
<td>6.4.6.5. Maximum Offline Interval</td>
<td>189</td>
</tr>
<tr>
<td>6.4.6.6. Workload Distribution</td>
<td>189</td>
</tr>
<tr>
<td>6.4.7. Controlling Assignment of Shards to Channels</td>
<td>191</td>
</tr>
<tr>
<td>6.4.8. Disk vs. Memory Parallel Queues</td>
<td>191</td>
</tr>
<tr>
<td>6.5. Batch Loading for Data Warehouses</td>
<td>193</td>
</tr>
<tr>
<td>6.5.1. How It Works</td>
<td>193</td>
</tr>
<tr>
<td>6.5.2. Important Limitations</td>
<td>193</td>
</tr>
<tr>
<td>6.5.3. Batch Applier Setup</td>
<td>194</td>
</tr>
<tr>
<td>6.5.4. JavaScript Batchloader Scripts</td>
<td>194</td>
</tr>
<tr>
<td>6.5.4.1. JavaScript Batchloader with Parallel Apply</td>
<td>195</td>
</tr>
<tr>
<td>6.5.5. Staging Tables</td>
<td>196</td>
</tr>
<tr>
<td>6.5.5.1. Staging Table Names</td>
<td>196</td>
</tr>
<tr>
<td>6.5.5.2. Whole Record Staging</td>
<td>196</td>
</tr>
<tr>
<td>6.5.5.3. Delete Key Staging</td>
<td>196</td>
</tr>
<tr>
<td>6.5.5.4. Staging Table Generation</td>
<td>197</td>
</tr>
<tr>
<td>6.5.6. Character Sets</td>
<td>197</td>
</tr>
<tr>
<td>6.5.7. Supported CSV Formats</td>
<td>197</td>
</tr>
<tr>
<td>6.5.8. Columns in Generated CSV Files</td>
<td>198</td>
</tr>
<tr>
<td>6.5.9. Batchloading Opcodes</td>
<td>198</td>
</tr>
<tr>
<td>6.5.10. Time Zones</td>
<td>199</td>
</tr>
<tr>
<td>6.5.11. Data File Partitioning</td>
<td>199</td>
</tr>
<tr>
<td>6.6. Deployment Security</td>
<td>199</td>
</tr>
<tr>
<td>6.6.1. Enabling Security</td>
<td>200</td>
</tr>
<tr>
<td>6.6.1.1. Enabling Security using the Staging Method</td>
<td>200</td>
</tr>
<tr>
<td>6.6.1.2. Enabling Security using the INI Method</td>
<td>201</td>
</tr>
<tr>
<td>6.6.2. Disabling Security</td>
<td>201</td>
</tr>
<tr>
<td>6.6.3. Creating Suitable Certificates</td>
<td>202</td>
</tr>
<tr>
<td>6.6.4. Installing from a Staging Host with Manually Generated Certificates</td>
<td>202</td>
</tr>
<tr>
<td>6.6.5. Installing via INI File with Manually Generated Certificates</td>
<td>202</td>
</tr>
<tr>
<td>6.6.6. Replacing the TLS Certificate from a Staging Directory</td>
<td>202</td>
</tr>
<tr>
<td>6.6.7. Removing TLS Encryption from a Staging Directory</td>
<td>203</td>
</tr>
<tr>
<td>6.6.8. Handling Database level Security</td>
<td>203</td>
</tr>
<tr>
<td>6.6.9. Creating the Truststore and Keystore</td>
<td>204</td>
</tr>
<tr>
<td>6.6.9.1. Creating Your Own Client and Server Certificates</td>
<td>204</td>
</tr>
<tr>
<td>6.6.9.2. Creating a Custom Certificate and Getting it Signed</td>
<td>206</td>
</tr>
<tr>
<td>6.6.9.3. Using an existing Certificate</td>
<td>207</td>
</tr>
</tbody>
</table>
List of Figures

2.1. Internals: MySQL Extraction .............................................................. 29
2.2. Internals: Amazon RDS MySQL, or Amazon Aurora, Offboard Extraction .............................................................. 29
2.3. Topologies: Master/Slave ................................................................. 31
2.4. Topologies: Multi-Master ................................................................. 31
2.5. Topologies: Fan-Out ....................................................................... 32
2.6. Topologies: Fan-In ...................................................................... 32
2.7. Topologies: Cluster-Slave ............................................................. 33
2.8. Topologies: Heterogeneous Operation ........................................... 34
3.1. Topologies: Master/Slave ............................................................. 36
3.2. Topologies: RDS Extraction ....................................................... 44
3.3. Fig 1. AWS Config ................................................................. 50
3.4. Fig 2. AWS Config ................................................................. 51
3.5. Fig 3. AWS Config ................................................................. 52
3.6. Fig 4. AWS Config ................................................................. 53
3.7. Fig 5. AWS Config ................................................................. 54
3.8. Fig 6. AWS Config ................................................................. 55
3.9. Fig 7. AWS Config ................................................................. 55
3.10. Topologies: Replicating Data Out of a Cluster .......................... 56
4.1. Topologies: MySQL to Oracle ................................................... 60
4.2. Topologies: Oracle to MySQL ................................................... 61
4.3. Topologies: Oracle to Oracle ..................................................... 62
4.4. Oracle Extraction with Synchronous CDC .................................. 63
4.5. Oracle Extraction with Asynchronous CDC ............................. 64
5.1. Topologies: Replicating to Cassandra ...................................... 84
5.2. Topologies: Replicating to Elasticsearch ..................................... 92
5.3. Topologies: Replicating to Hadoop ........................................... 102
5.4. Topologies: Hadoop Replication Operation ................................ 103
5.5. Topologies: Replicating to Kafka ............................................. 113
5.6. Topologies: Replicating to MongoDB ...................................... 120
5.7. Topologies: Replicating to MySQL .......................................... 131
5.8. Topologies: Replicating to Oracle ............................................ 140
5.9. Topologies: Replicating to Amazon Redshift ............................ 145
5.10. Topologies: Redshift Replication Operation ............................ 146
5.11. Topologies: Replicating to Vertica .......................................... 156
6.1. Topologies: Fan-in ................................................................. 164
6.2. Topologies: Replicating into a Dataservice .............................. 174
6.3. Batchloading: JavaScript ......................................................... 195
7.1. Parallel Extractor: Extraction Sequence ................................... 228
7.2. Parallel Extractor: Extraction Operation .................................. 228
7.3. Cacti Monitoring: Example Graphs ........................................... 239
9.1. tpm Staging Based Deployment ................................................ 316
9.2. tpm INI Based Deployment ..................................................... 317
10.1. Filters: Pipeline Stages on Masters ........................................ 405
10.2. Filters: Pipeline Stages on Slaves ........................................... 406
B.1. Tungsten Deployment .......................................................... 466
List of Tables

1. Supported Extractors ............................................................................................................. 18
2. Supported Appliers .................................................................................................................. 18
2.1. Key Terminology .................................................................................................................. 22
4.1. Data Type differences when replicating data from MySQL to Oracle .................................. 64
4.2. Data Type Differences when Replicating from Oracle to MySQL or Oracle ....................... 65
4.3. setupCDC.conf Configuration File Parameters ...................................................................... 73
5.1. Optional Elasticsearch Applier Properties ........................................................................... 96
5.2. Hadoop Replication Directory Locations ............................................................................ 104
5.3. Optional Kafka Applier Properties ...................................................................................... 117
5.4. Data Type differences when replicating data from MySQL to Oracle ................................ 140
6.1. Continennt Tungsten Directory Structure .......................................................................... 197
8.1. check_tungsten_latency Options ......................................................................................... 243
8.2. check_tungsten_online Options ......................................................................................... 244
8.3. check_tungsten_services Options ....................................................................................... 244
8.4. dlisct Command-line Options ............................................................................................ 246
8.5. dlisct Supported Templates .................................................................................................. 247
8.6. dsctl Commands ................................................................................................................ 257
8.7. dsctl Command-line Options ............................................................................................. 258
8.8. dsctl Command-line Options ............................................................................................. 258
8.9. dsctl Command-line Options ............................................................................................. 258
8.10. multi_trepctl Command-line Options .............................................................................. 261
8.11. multi_trepctl --output Option .......................................................................................... 262
8.12. multi_trepctl Commands .................................................................................................. 263
8.13. query Common Options ..................................................................................................... 265
8.14. replicator Commands ......................................................................................................... 265
8.15. replicator Commands Options for condrestart ................................................................. 266
8.16. replicator Commands Options for console ....................................................................... 266
8.17. replicator Commands Options for restart ......................................................................... 266
8.18. replicator Commands Options for start ............................................................................ 267
8.19. setupCDC.conf Configuration Options ............................................................................ 267
8.20. tli Options ........................................................................................................................ 271
8.21. trepctl Command-line Options ......................................................................................... 279
8.22. trepctl Replicator Wide Commands ................................................................................... 280
8.23. trepctl Service Commands ................................................................................................ 282
8.24. trepctl backup Command Options ...................................................................................... 283
8.25. trepctl clients Command Options ...................................................................................... 285
8.26. trepctl offline-deferred Command Options ....................................................................... 287
8.27. trepctl online Command Options ...................................................................................... 289
8.28. trepctl purge Command Options ....................................................................................... 293
8.29. trepctl reset Command Options ......................................................................................... 295
8.30. trepctl setrole Command Options ..................................................................................... 296
8.31. trepctl shard Command Options ....................................................................................... 296
8.32. trepctl status Command Options ....................................................................................... 297
8.33. trepctl wait Command Options .......................................................................................... 304
8.34. tpswsw Common Options .................................................................................................... 305
8.35. tungsten_provision_slave Command-line Options ............................................................. 310
8.36. tungsten_read_master_events Command-line Options ................................................. 311
8.37. tungsten_send_diag Command-line Options ...................................................................... 312
8.38. tungsten_set_position Command-line Options ................................................................... 312
9.1. TPM Deployment Methods .................................................................................................. 317
9.2. tpm Core Options ................................................................................................................ 325
9.3. tpm Commands .................................................................................................................... 326
9.4. tpm Common Options .......................................................................................................... 333
9.5. tpm Validation Checks .......................................................................................................... 336
9.6. tpm Configuration Options .................................................................................................. 354
D.1. Continuent Tungsten Directory Structure ........................................................................... 493
D.2. Continuent Tungsten tungsten Sub-Directory Structure ......................................................... 500
E.1. THL Event Format ................................................................................................................ 503
Preface

This manual documents Tungsten Clustering 5.4 up to and including 5.4.1 build 41. Differences between minor versions are highlighted stating the explicit minor release version, such as 5.4.1.x.

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

1. Legal Notice

The trademarks, logos, and service marks in this Document are the property of Continuent or other third parties. You are not permitted to use these Marks without the prior written consent of Continuent or such appropriate third party. Continuent, Tungsten, uni/cluster, m/cluster, p/cluster, uc/connector, and the Continuent logo are trademarks or registered trademarks of Continuent in the United States, France, Finland and other countries.

All Materials on this Document are [and shall continue to be] owned exclusively by Continuent or other respective third party owners and are protected under applicable copyrights, patents, trademarks, trade dress and/or other proprietary rights. Under no circumstances will you acquire any ownership rights or other interest in any Materials by or through your access or use of the Materials. All right, title and interest not expressly granted is reserved to Continuent.

All rights reserved.

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 [382]` 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 formatted using this style. 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 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/staging
shell> unzip tungsten-replicator-5.4.1-41.zip
```

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, or `{Cmd.exe}` or `{PowerShell}` on Windows.

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.
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 Master/Slave deployment?
  - Are you replicating to or from Oracle?
  - Are you trying to configure Hadoop replication?
  - Are you trying to configure Vertica replication?
  - Are you trying to configure Elasticsearch replication?
  - Are you trying to configure Kafka replication?
  - Are you trying to configure Cassandra replication?
- 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 9.5.14, “tpm update Command”.
- Are you trying to update or change the configuration of your system?
  See Section 9.5.14, “tpm update Command”.
- Would you like to perform database or operating system maintenance?
  See Section 7.14, “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 and Google Cloud SQL, and applied to a variety of transactional stores, NoSQL stores and datawarehouse stores. In addition, the replicator can extract from Oracle via CDC, available for Oracle versions up to and including 11g. 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 multi-master, and can be used efficiently in cross-site deployments.

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

Features in Tungsten Replicator from version 5.2 onwards:

- 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, Elasticsearch, Apache Cassandra and Apache Kafka
- Includes support for replicating to and from Amazon RDS (MySQL) deployments
- 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

<table>
<thead>
<tr>
<th>Source</th>
<th>5.3</th>
<th>5.4</th>
<th>6.0</th>
<th>6.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL (5.0 to 5.6)</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>MySQL 5.7</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>MySQL 8</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>MariaDB [5.5, 10]</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Amazon Aurora/RDS MySQL</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Google Cloud MySQL</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Microsoft Azure</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

### Table 1.2. Supported Appliers

<table>
<thead>
<tr>
<th>Target</th>
<th>5.3</th>
<th>5.4</th>
<th>6.0</th>
<th>6.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL (incl MariaDB)</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Amazon Aurora/RDS MySQL</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Microsoft Azure</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Google Cloud MySQL</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Oracle (incl. Cloud)</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>PostgreSQL (incl. Cloud)</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Hadoop</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vertica</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Amazon Redshift</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MongoDB</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MongoDB Atlas</td>
<td></td>
<td></td>
<td></td>
<td>6.1.3</td>
</tr>
</tbody>
</table>

18
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 multi-master, 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, Oracle, and Vertica, among others.

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 master/slave are easy to understand with the basic concepts described above. More complex topologies use the same core components. For example, multi-master 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.

• Directly from Oracle up to Oracle 11g using the Oracle CDC system. The Change Data Capture solution within Oracle makes it possible to extract information from the tables within Oracle, although it’s availability is limited and implies some overhead on the Oracle service. It can provide a flexible solution for extracting row-based information.

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:

• MongoDB
• MySQL, including Amazon RDS targets
• Oracle
• HPE Vertica
• Hadoop, compatible with all major distributions
• Amazon RedShift
• Apache Kafka
• Apache Cassandra
• Elasticsearch

For more information on how the heterogeneous replicator works, see Section 2.9.1, “How Heterogeneous Replication Works”. For more information on the batch applier, which works with datawarehouse targets, see Section 6.5, “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 multi-master, 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 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 10, Replication Filters.
Chapter 2. Deployment Overview

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

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

### 2.1. Deployment Sources

Tungsten Clustering 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 *tpm* Methods” for more information on the available installation methods.

<table>
<thead>
<tr>
<th>Deployment Type/Package</th>
<th>TAR/GZip</th>
<th>RPM/DEB</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>tpm</em> Command-line Configuration</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><em>tpm</em> INI File Configuration</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Deploy Entire Cluster</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Deploy Per Machine</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

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

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

`tungsten-replicator-5.4.1-41.tar.gz`
Deployment Overview

The version number is **5.4.1** and build number **41**. 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
$ tar zxf tungsten-replicator-5.4.1-41.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-5.4.1-41
```

### 2.1.2. Using the RPM and DEB package files

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

**Installation**

Installing the RPM or DEB 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/DEB 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:

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

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

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

To obtain the package files, you can use one of the following methods:

- Download from an existing download page.
- For **yum** platforms (RHEL/CentOS/Amazon Linux), add the package source to your **yum** configuration. For the current stable (GA) release packages:
  ```shell
  $ rpm -i http://releases.continuent.com.s3.amazonaws.com/replicator-release-stable-0.0-1.x86_64.rpm
  ```
  For nightly builds:
  ```shell
  $ rpm -i http://releases.continuent.com.s3.amazonaws.com/replicator-release-nightly-0.0-1.x86_64.rpm
  ```
- For Ubuntu/Debian packages:
Deployment Overview

```bash
root-shell> echo "deb http://apt.tungsten-replicator.org/ stable main" \\
> /etc/apt/sources.list.d/tungsten_stable.list
```

Nightly builds are also available:

```bash
root-shell> echo "deb http://apt-nightly.tungsten-replicator.org/ nightly main" \\
> /etc/apt/sources.list.d/tungsten_nightly.list
```

Then update your **apt** repository:

```bash
root-shell> apt-get update
```

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

- On RHEL/CentOS/Amazon Linux:
  ```bash
  root-shell> yum install tungsten-replicator
  ```
- On Ubuntu/Debian:
  ```bash
  root-shell> apt-get install tungsten-replicator
  ```

### Upgrades

If you upgrade to a new version of the RPM or DEB 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 Methods”.
- Standardize the OS and database prerequisites. There are Puppet and Chef modules available for immediate use or as a template for modifications.
- 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 Clustering. For more information on the network ports required for Tungsten Clustering operation, see Section B.3.2.1, “Network Ports”.
  - If possible, use authentication and SSL connectivity between hosts to protect your data and authorisation for the tools used in your deployment.
    
    See Section 6.6, “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 slave servers.
  - Make sure you know the structure of the Tungsten Clustering home directory and how to initialize your environment for administration. See Section 7.1, “The Tungsten Clustering Home Directory” and Section 7.2, “Establishing the Shell Environment”.
  - Prior to migrating applications to Tungsten Clustering test failover and recovery procedures from Chapter 7, *Operations Guide*. Be sure to try recovering a failed master and reprovisioning failed slaves.
2.2.2. Best Practices: Operations

- Setup proper monitoring for all servers as described in Section 7.16, "Monitoring Tungsten Clustering".
- Configure the Tungsten Clustering services to startup and shutdown along with the server. See Section 2.6, "Configuring Startup on Boot".

2.2.3. 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 7.14.3, "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. Prepare Hosts

Using Puppet is the fastest way to prepare a host for Tungsten Clustering. These instructions will show you how to install Puppet and prepare a host to run Tungsten Clustering. If you want to prepare the hosts without Puppet, follow the guidelines in Appendix B, Prerequisites.

- Make sure Puppet and all required packages are installed. See https://docs.puppetlabs.com/guides/puppetlabs_package_repositories.html if you have any issues getting Puppet installed.

For RHEL/CentOS-based distributions:

```bash
shell > rpm -ivh http://yum.puppetlabs.com/puppetlabs-release-el-6.noarch.rpm
shell > yum install -y ruby rubygems ruby-devel puppet
```

For Ubuntu-based distributions:

```bash
shell > apt-get update
shell > apt-get install -y ruby ruby-dev puppet
```

- Install the Continuent Puppet module.

```bash
shell > mkdir -p /etc/puppet/modules
shell > puppet module install continuent/tungsten
```

- If you do not have DNS entries for the hosts in use, update the `/etc/hosts` file so that it reflects the proper IP addresses and complete hostname.

```bash
shell > puppet apply -e "host { 'db1.west.example.com': ip => '192.168.11.101', }
host { 'db2.west.example.com': ip => '192.168.11.102', }
host { 'db3.west.example.com': ip => '192.168.11.103', }
"
```

2.3.1. Prepare MySQL Hosts

Use the Continuent Puppet module to install all prerequisites including MySQL. This will implement the prerequisites described in Section B.3, "Host Configuration" and Section B.4, "MySQL Database Setup".

```bash
shell > puppet apply -e "class { 'tungsten' :
  installMysql => true,
  replicationUser => 'tungsten',
  replicationPassword => 'secret',
  appUser => 'app_user',
  appPassword => 'secret',
}"
```

2.3.2. Deploy SSH Keys

The `tpm` script uses SSH to execute commands on each host. There are two simple ways to install these keys.

- Provide the SSH certificate and key to Puppet. In each of the examples below you may include an SSH certificate and key that will be assigned to the `tungsten` system user.

```bash
shell > puppet apply -e "class { 'tungsten' :
  ```
2.4. Common tpm Options During Deployment

There are a variety of tpm options that can be used to control or 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 9.5.14, “`tpm update Command`” for more information on using `tpm update`.

- `--datasource-systemctl-service` [375]

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` [375] 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 start
```

Different commands must follow the same basic structure, the command configured by `--datasource-systemctl-service` [375], the service-name, and the status (i.e. `stop`).

### 2.5. 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` [212] state:

```
shell> replicator start offline
```

In a clustered environment, if the cluster was configured with `auto-enable=false` [365] then you will need to put each node online individually.

### 2.6. 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
```

### 2.7. 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 2.7.1, “Removing a Datasource from an Existing Deployment”

### 2.7.1. 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
       ```

     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`:
     a. Remove the INI configuration file:

       ```shell
       rm /etc/tungsten/tungsten.ini
       ```
   2. Remove the installed software directory:

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

### 2.8. 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

#### 2.8.1. Tungsten Replicator Extraction Operation

<table>
<thead>
<tr>
<th>Replication Operation Support</th>
<th>Statements Replicated</th>
<th>Rows Replicated</th>
<th>Schema Replicated</th>
<th>ddlscan Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support</td>
<td>Yes, within MySQL/MySQL Topologies only</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes, supported for mixed MySQL, and data warehouse targets</td>
</tr>
</tbody>
</table>

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 its different formats, such as the statement, row and mixed-based logging.
- Remotely from the MySQL server over a network, including reading from an Amazon RDS or 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
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

### 2.8.2. 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-Slave 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

2.8.3. Understanding Deployment Topologies

There are a number of different methods in which Tungsten Replicator can be configured, review Section 2.8.2, “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 2.8.3.1, “Simple Master/Slave Topology”

A simple Master/Slave topology replicating from one source host to one target.

• Section 2.8.3.2, “Multi-Master Topology”

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

This topology can only be configured between MySQL hosts.

• Section 2.8.3.3, “Fan-Out Topology”

A more advanced Master/Slave 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 2.8.3.4, “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 example, remap schemas.

• Section 2.8.3.5, “Replicating in/out of an existing Tungsten Cluster”

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

2.8.3.1. Simple Master/Slave Topology

Master/slave 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 master/slave topology is that changes in the master are distributed and applied to the each of the configured slaves.
2.8.3.2. Multi-Master Topology

A multi-master topology, relies on a number of individual services that are used to define a master/slave topology between each group of hosts. In a three-node multimaster setup, for example, three different services are created on each host, each service creates a master/slave relationship between a primary host (itself) and the remote slaves. A change on any individual host will be replicated to the other databases in the topology creating the multi-master configuration.
2.8.3.3. 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

2.8.3.4. Fan-In Topology

The fan-in topology is the logical opposite of a master/slave topology. In a fan-in topology, the data from two (or more) masters is combined together on one slave. 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

2.8.3.5. 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 master/slave 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.
2.9. 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
  - ...HP Vertica
  - ...Elasticsearch
  - ...Apache Kafka
  - ...Apache Cassandra
  - ...MongoDB

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

2.9.1. 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.
2.9.1.1. JDBC Applier based Replication

Replication into targets where the JDBC Driver can be used, such as Oracle and Postgres, work as shown in Figure 2.8, “Topologies: Heterogeneous Operation”.

Figure 2.8. Topologies: Heterogeneous Operation

The process works as follows:

1. Data is extracted from the source database. The exact method depends on whether data is being extracted from MySQL or Oracle.

   - For MySQL:
     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) appended to the metadata of the THL event.

   - For Oracle CDC:
     The Oracle Change Data Capture (CDC) system records the row-level changes made to a table into a change table. Tungsten Replicator reads the change information from the change tables and generates row-based transactions within the THL.
Deployment Overview

In both cases, 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 the two databases 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 master to the slave.

3. On the slave [or 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.

2.9.1.2. Native Applier Replication [e.g. MongoDB, Elasticsearch]

For heterogeneous replication where data is written into a target database using a native applier, such as MongoDB or Elasticsearch, 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.

2.9.1.3. 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 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) for each primary key) are then inserted.

2.9.1.4. 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 ddlsan 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 ddlsan is not required.
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.6, “Pre-requisite Checklist”

3.2. Deploying a Master/Slave Topology

Master/slave 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 master/slave topology is that changes in the master are distributed and applied to the each of the configured slaves.

Figure 3.1. Topologies: Master/Slave

This deployment style can be used against the following sources

- MySQL Community Edition
- MySQL Enterprise Edition
- Percona MySQL
- MariaDB
Deploying MySQL Extractors

- 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 RDS/Amazon Aurora”

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

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

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

```
$ cd /opt/continuent/software
$ tar zxf tungsten-replicator-5.4.1-41.tar.gz
```

- Change to the Tungsten Replicator staging directory:

```
$ cd tungsten-replicator-5.4.1-41
```

- Onboard Installation

Configure the repliactor 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

```
$ ./tools/tpm configure defaults
   --reset
   --install-directory=/opt/continuent
   --user=tungsten
   --profile-script=~/.bash_profile
   --mysql-allow-intensive-checks=true
   --disable-security-controls=true

$ ./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

$ vi /etc/tungsten/tungsten.ini
```

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 [397]
reset [397]

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

- --install-directory=/opt/continuent [382]
  
  install-directory=/opt/continuent [382]

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 [403]
  
  user=tungsten [403]

System User

- --profile-script=~/.bash_profile [394]
  
  profile-script=~/.bash_profile [394]

Append commands to include env.sh in this profile script

- --mysql-allow-intensive-checks=true [389]
  
  mysql-allow-intensive-checks=true [389]

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 [378]
  
  disable-security-controls=true [378]

Disables all forms of security, including SSL, TLS and authentication

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 [386]
  
  master=localhost [386]

The hostname of the master (extractor) within the current service. If the current host does not match this specification, then the deployment will by default be configured as a master/extractor.

- --members=localhost [386]
  
  members=localhost [386]

Hostnames for the dataservice members

- --enable-heterogeneous-service=true [380]
  
  enable-heterogeneous-service=true [380]

On a Master

- --mysql-use-bytes-for-string [391] 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-dbs 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 Slave
Deploying MySQL Extractors

- **--mysql-use-bytes-for-string** is set to true.

- **pkey** filter is enabled (q-to-dbms stage).

- **--replication-port=3306**

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

- **--replication-user=tungsten_alpha**

  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**

  The password to be used when connecting to the database using the corresponding **--replication-user**.

- **--datasource-mysql-conf=/etc/my.cnf**

  MySQL config file

  In the above example, **--datasource-mysql-conf**, 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 \n  --install-directory=/opt/continuent \n  --user=tungsten \n  --profile-script=~/.bash_profile \n  --mysql-allow-intensive-checks=true \n  --skip-validation-check=MySQLAvailableCheck \n  --skip-validation-check=MySQLConfFile \n  --disable-security-controls=true
  ```

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

  ```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
  ```
Deploying MySQL Extractors

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>skip-validation-check</td>
<td>MySQLAvailableCheck</td>
</tr>
<tr>
<td>disable-security-controls</td>
<td>true</td>
</tr>
<tr>
<td>[alpha] master</td>
<td>offboardhost</td>
</tr>
<tr>
<td>members</td>
<td>offboardhost</td>
</tr>
<tr>
<td>enable-heterogeneous-service</td>
<td>true</td>
</tr>
<tr>
<td>privileged-master</td>
<td>true</td>
</tr>
<tr>
<td>replication-host</td>
<td>dbhost</td>
</tr>
<tr>
<td>replication-port</td>
<td>3306</td>
</tr>
<tr>
<td>replication-user</td>
<td>tungsten_alpha</td>
</tr>
<tr>
<td>replication-password</td>
<td>secret</td>
</tr>
</tbody>
</table>

**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** [397]
  
  `reset` [397]

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

- **--install-directory=/opt/continuent** [382]
  
  `install-directory=/opt/continuent` [382]

  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** [403]
  
  `user=tungsten` [403]

  System User

- **--profile-script=~/.bash_profile** [394]
  
  `profile-script=~/.bash_profile` [394]

  Append commands to include env.sh in this profile script

- **--mysql-allow-intensive-checks=true** [389]
  
  `mysql-allow-intensive-checks=true` [389]

  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** [335]
  
  `skip-validation-check=MySQLAvailableCheck` [335]

  The **--skip-validation-check** [335] 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: The datasource root@centos:3306 (WITH PASSWORD) uses MyISAM as the default storage engine (MySQLDefaultTableTypeCheck).
```

The check in this case is `MySQLDefaultTableTypeCheck` [346], and could be ignored using **--skip-validation-check=MySQLDefaultTableTypeCheck** [335].

Setting both **--skip-validation-check** [335] and **--enable-validation-check** [333] is equivalent to explicitly disabling the specified check.
• `--skip-validation-check=MySQLConfFile`

The `--skip-validation-check` 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`, and could be ignored using `--skip-validation-check=MySQLDefaultTableTypeCheck`.

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

• `--skip-validation-check=RowBasedBinaryLoggingCheck`

The `--skip-validation-check` 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`, and could be ignored using `--skip-validation-check=MySQLDefaultTableTypeCheck`.

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

• `--disable-security-controls=true`

Disables all forms of security, including SSL, TLS and authentication

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`

The hostname of the master (extractor) within the current service. If the current host does not match this specification, then the deployment will by default be configured as a master/extractor.

• `--members=offboardhost`

Hostnames for the dataservice members
Deploying MySQL Extractors

- `--enable-heterogeneous-service=true` [380]
  
- **On a Master**
  - `--mysql-use-bytes-for-string` [391] 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 Slave**
  - `--mysql-use-bytes-for-string` [391] is set to true.
  - `pkey` filter is enabled (in the `q-to-dbms` stage).
  - `--privileged-master=true` [394]
    
- Does the login for the master database service have superuser privileges

- `--replication-host=dbhost` [396]
  
- 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 slave (relay) connection.

- `--replication-port=3306` [396]
  
- The network port used to connect to the database server. The default port used depends on the database being configured.

- `--replication-user=tungsten_alpha` [397]
  
- 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` [396]
  
- The password to be used when connecting to the database using the corresponding `--replication-user`.

- 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` [380], 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 slave service following the relevant step within Chapter 5, Deploying Appliers.

  Following installation of the applier, the services can be started. For information on starting and stopping Tungsten Clustering see Section 2.5, "Starting and Stopping Tungsten Replicator"; configuring init scripts to startup and shutdown when the system boots and shuts down, see Section 2.6, "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`:

```bash
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 master:

```bash
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/
masterPort             : 10000
maximumStoredSeqNo     : 3593
minimumStoredSeqNo     : 0
offlineRequests        : NONE
pendingError           : NONE
pendingErrorCode       : NONE
pendingErrorEventId    : NONE
pendingErrorSeqno      : -1
pendingExceptionMessage: NONE
pipelineSource         : jdbc:mysql:thin://host1:3306/
relativeLatency        : 604984.598
resourcePrecedence     : 99
rmiPort                : 10098
role                   : master
seqnoType              : java.lang.Long
serviceName            : alpha
serviceType            : local
simpleServiceName      : alpha
siteName               : default
sourceId               : host1
state                  : ONLINE
timeInStateSeconds     : 604983.621
transitioningTo        :
uptimeSeconds          : 1260173.328
version                : Tungsten Replicator 5.4.1 build 41
Finished status command...
```

For more information on using `trepctl`, see Section 8.18, “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 RDS/Amazon Aurora

Replicating from Amazon RDS (or Amazon Aurora), operates by directly accessing the binary log provided by RDS/Aurora and enables you to take advantage of the Amazon Web, either replicating from the remote RDS/Aurora instance, or to a standard EC2 instance within AWS. The complexity with RDS/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 RDS without requiring \texttt{SUPER} privileges, which are unavailable within an RDS 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: RDS 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 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 and DEB package files"], or download the compressed tarball and unpack it, either on the source host, or on the staging host:

  ```sh
cd /opt/continuent/software
  tar zxf tungsten-replicator-5.4.1-41.tar.gz
  ```

- Change to the Tungsten Replicator staging directory:

  ```sh
cd tungsten-replicator-5.4.1-41
  ```

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

  ```sh
  ./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 --disable-security-controls=true
  ./tools/tpm configure alpha --master=localhost
  ```

Show Staging

Show INI

```sh
```

44
Deploying MySQL Extractors

```
--members=localhost \  
--enable-heterogeneous-service=true \  
--replication-hosts=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
disable-security-controls=true

[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 hide this detail:

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

- **--reset [397]**
  
  reset [397]

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

- **--install-directory=/opt/continuent [382]**
  
  install-directory=/opt/continuent [382]

  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 [403]**
  
  user=tungsten [403]

  System User

- **--profile-script=/.bash_profile [394]**
  
  profile-script=/.bash_profile [394]

  Append commands to include env.sh in this profile script

- **--mysql-allow-intensive-checks=true [389]**
  
  mysql-allow-intensive-checks=true [389]

  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.
Deploying MySQL Extractors

```bash
skip-validation-check=InstallerMasterSlaveCheck [335]
```

The `--skip-validation-check [335]` 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:

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

The check in this case is `MySQLDefaultTableTypeCheck [346]`, and could be ignored using `--skip-validation-check=MySQLDefaultTableTypeCheck [335]`.

Setting both `--skip-validation-check [335]` and `--enable-validation-check [333]` is equivalent to explicitly disabling the specified check.

- `--skip-validation-check=MySQLPermissionsCheck [335]`
- `skip-validation-check=MySQLPermissionsCheck [335]`

The `--skip-validation-check [335]` 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:

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

The check in this case is `MySQLDefaultTableTypeCheck [346]`, and could be ignored using `--skip-validation-check=MySQLDefaultTableTypeCheck [335]`.

Setting both `--skip-validation-check [335]` and `--enable-validation-check [333]` is equivalent to explicitly disabling the specified check.

- `--skip-validation-check=MySQLBinaryLogsEnabledCheck [335]`
- `skip-validation-check=MySQLBinaryLogsEnabledCheck [335]`

The `--skip-validation-check [335]` 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:

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

The check in this case is `MySQLDefaultTableTypeCheck [346]`, and could be ignored using `--skip-validation-check=MySQLDefaultTableTypeCheck [335]`.

Setting both `--skip-validation-check [335]` and `--enable-validation-check [333]` is equivalent to explicitly disabling the specified check.

- `--skip-validation-check=MySQLMyISAMCheck [335]`

46
skip-validation-check=MySQLMyISAMCheck

The `--skip-validation-check` 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`, and could be ignored using `--skip-validation-check=MySQLDefaultTableTypeCheck`.

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

- `--disable-security-controls=true`

Disables all forms of security, including SSL, TLS and authentication

**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`

  The hostname of the master [extractor] within the current service. If the current host does not match this specification, then the deployment will by default be configured as a master/extractor.

- `--members=localhost`

  Hostnames for the dataservice members

- `--enable-heterogeneous-service=true`
Deploying MySQL Extractors

enable-heterogeneous-service=true

• On a Master
  • --mysql-use-bytes-for-string 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-dbs 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 Slave
  • --mysql-use-bytes-for-string is set to true.
  • pkey filter is enabled (q-to-dbms stage).

--privileged-master=false

privileged-master=false

Does the login for the master database service have superuser privileges

• --replication-host=rds.endpoint.url
  replication-host=rds.endpoint.url
  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 slave [relay] connection.

• --replication-port=3306
  replication-port=3306
  The network port used to connect to the database server. The default port used depends on the database being configured.

• --replication-user=tungsten_alpha
  replication-user=tungsten_alpha
  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
  replication-password=secret
  The password to be used when connecting to the database using the corresponding --replication-user.

• --datasource-mysql-conf=/dev/null
  datasource-mysql-conf=/dev/null
  MySQL config file

• --svc-extractor-filters=dropcatalogdata
  svc-extractor-filters=dropcatalogdata
  Replication service extractor filters

• --property=replicator.service.comments=true
  property=replicator.service.comments=true
  The --property option enables you to explicitly set property values in the target files. A number of different models are supported:

  • key=value
Deploying MySQL Extractors

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

  In the above examples,

  - `enable-heterogenous-service [380]`, is only required if the target applier is NOT a MySQL database
  - `datasource-mysql-conf [373]`, 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 slave service following the relevant step within Chapter 5, Deploying Appliers.

  Following installation of the applier, the services can be started. For information on starting and stopping Tungsten Clustering see Section 2.5, "Starting and Stopping Tungsten Replicator"; configuring init scripts to startup and shutdown when the system boots and shuts down, see Section 2.6, "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:

```sql
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:

```bash
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`:

```bash
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:

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

   Where `repgroup` is the replicator group name.

2. Set the new parameter value:

   ```bash
   shell> rds-modify-db-parameter-group repgroup --parameters \n   "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

   **Amazon RDS**
   - Dashboard
   - Databases
   - Query Editor
   - Performance Insights
   - Snapshots
   - Automated backups
   - Reserved instances
   - Security groups
   - Subnet groups
   - **Parameter groups**
   - Option groups
   - Events
   - Event subscriptions
   - Recommendations

2. Select the “Create Parameter Group” Button to the top right
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 Parameter group and a DB Cluster Parameter Group.
Figure 3.5. Fig 3. AWS Config
### Parameter group details

To create a parameter group, choose a parameter group family, then name and describe.

<table>
<thead>
<tr>
<th>Parameter group family</th>
<th>DB family that this DB parameter group will apply to</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>aurora-mysql5.7</strong></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DB Parameter Group</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Group name</th>
<th>Identifier for the DB parameter group</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>mynewparametergroup</strong></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>Description for the DB parameter group</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>New Parameters for Tungsten Replicator</strong></td>
<td></td>
</tr>
</tbody>
</table>
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.
Deploying MySQL Extractors

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 master/slave 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.
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-slave server. That server will download THL data and apply them to the local server. If the cluster-slave has more than one server; one of them will be declared the relay (or master). The other members of the cluster-slave may also download THL data from that server.

If the relay for the cluster-slave 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 master, slaves 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 \-enable-heterogeneous-service=true [380] prior to beginning. The same option must be included when installing the Tungsten Replicator.

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-slave 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-5.4.1-41.tar.gz
```

2. Change to the unpackaged directory:

```
shell> cd tungsten-replicator-5.4.1-41
```

3. Configure the replicator

   Click the link below to switch examples between Staging and INI methods
   Show Staging
Deploying MySQL Extractors

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

shell> ./tools/tpm configure alpha \\
   --master=host1 \\
   --slave=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

[alpha]
master=host1
slave=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 [382]

   install-directory=/opt/continuent [382]

   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 [394]

   profile-script=~/.bash_profile [394]

   Append commands to include env.sh in this profile script

• --replication-password=secret [396]

   replication-password=secret [396]

   The password to be used when connecting to the database using the corresponding --replication-user [397].

• --replication-port=13306 [396]

   replication-port=13306 [396]

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

• --replication-user=tungsten [397]

   replication-user=tungsten [397]
For databases that required authentication, the username to use when connecting to the database using the corresponding connection method (native, JDBC, etc.).

• --user=tungsten [403]
  user=tungsten [403]

System User

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 [386]
  master=host1 [386]

  The hostname of the master (extractor) within the current service. If the current host does not match this specification, then the deployment will by default be configured as a master/extractor.

• --slaves=host2,host3 [398]
  slaves=host2,host3 [398]

  What are the slaves for this dataservice?

• --thl-port=2112 [403]
  thl-port=2112 [403]

  Port to use for THL Operations

• --topology=cluster-alias [403]
  topology=cluster-alias [403]

  Replication topology for the dataservice Valid values are star, cluster-slave, master-slave, fan-in, clustered, cluster-alias, all-masters, direct

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 [386]
  relay=host6 [386]

  The hostname of the master (extractor) within the current service. If the current host does not match this specification, then the deployment will by default be configured as a master/extractor.

• --relay-source=alpha [396]
  relay-source=alpha [396]

  Dataservice name to use as a relay source

• --topology=cluster-slave [403]
  topology=cluster-slave [403]

  Replication topology for the dataservice Valid values are star, cluster-slave, master-slave, fan-in, clustered, cluster-alias, all-masters, direct

Important

If you are replicating to a non-MySQL server. Include the --enable-heterogeneous-service=true [380] option in the above command.
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 if you are taking over for MySQL native replication. See 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 Oracle Extractors

4.1. Deploying Oracle Replication using CDC

<table>
<thead>
<tr>
<th>Replication Operation Support</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Statements Replicated</td>
<td>No</td>
</tr>
<tr>
<td>Rows Replicated</td>
<td>Yes</td>
</tr>
<tr>
<td>Schema Replicated</td>
<td>No</td>
</tr>
<tr>
<td><code>ddiscan</code> Supported</td>
<td>Yes, for mixed Oracle/MySQL</td>
</tr>
</tbody>
</table>

Tungsten Clustering supports replication to and from Oracle as a datasource, and therefore also supports replication between Oracle databases. This allows replication of data from Oracle to other database appliers, including MySQL. CDC Replication is supported from Oracle 10g and 11g. See the Database Support prerequisites for more details.

Three variations of Oracle-based replication are officially supported:

- MySQL to Oracle

Figure 4.1. Topologies: MySQL to Oracle

For configuration, see Chapter 3, Deploying MySQL Extractors to configure the extractor and Section 5.7, "Deploying the Oracle Applier" to configure the applier.

- Oracle to MySQL
Figure 4.2. Topologies: Oracle to MySQL

For configuration, see Section 4.1.3, “Creating an Oracle to MySQL Deployment”

- Oracle to Oracle
Deploying Oracle Extractors

Figure 4.3. Topologies: Oracle to Oracle

For configuration, see Section 4.1.4, “Creating an Oracle to Oracle Deployment”

Replication in these configurations operates using two separate replicators:

- Replicator on the master extracts the information from the source database into THL.
- Replicator on the slave reads the information from the remote replicator as THL, and applies that to the target database.

4.1.1. How Oracle Extraction Works

When replicating to Oracle, row data extracted from the source database is applied to the target database as an Oracle database user using SQL statements to insert the row based data. A combination of the applier class for Oracle, and filters, are used to format the row events into suitable statements.

When replicating from Oracle, changes to the database are extracted using the Oracle Change Data Capture (CDC) system. Support is available for using Synchronous and Asynchronous CDC according to the version of Oracle that is being used:

<table>
<thead>
<tr>
<th>Edition</th>
<th>Synchronous CDC</th>
<th>Asynchronous CDC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Edition (SE)</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Enterprise Edition (EE)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Standard Edition 1 (SE1)</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Express Edition (XE)</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Both CDC types operate through a series of change tables. The change tables are accessed by subscribers which read information from the change tables to capture the change data. The method for populating the change tables depends on the CDC method:
• Synchronous CDC

Figure 4.4. Oracle Extraction with Synchronous CDC

Within Synchronous CDC, triggers are created on the source tables which are configured to record the change information into the change tables. Subscribers to the change tables then read the information. With Tungsten Replicator, the replicator acts as the subscriber, reads the change information and populates the change data into the THL used by the replicator. Because the information is extracted from the tables being updated using triggers, there is an overhead for the Synchronous CDC mode for all database operations while the triggers are executed.

In addition, because the changes are captured within the transaction boundary, the information is exposed within the CDC tables quicker. The synchronous CDC can therefore be quicker than asynchronous CDC.

• Asynchronous CDC
With Asynchronous CDC, information is taken from the Oracle redo logs and placed into the change tables. These changes are dependent on the supplemental logging enabled on the source database. Supplemental logging adds redo logging overhead, which increases the redo log size and management requirements. Tungsten Replicator uses Asynchronous HotLog mode, which reads information from the Redo logs and writes the changes into the change data tables.

In both solutions, Tungsten Replicator reads the change data generated by the Oracle CDC system in the CDC table. The change data is extracted from these tables and then written into THL so that it can be transferred to another replicator and applied to another supported database.

Note
More information on Oracle CDC can be found within the Oracle documentation.

### 4.1.2. Data Type Differences and Limitations

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 4.1, "Data Type differences when replicating data from MySQL to Oracle".

<table>
<thead>
<tr>
<th>MySQL Datatype</th>
<th>Oracle Datatype</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT</td>
<td>NUMBER(10, 0)</td>
<td></td>
</tr>
<tr>
<td>BIGINT</td>
<td>NUMBER(19, 0)</td>
<td></td>
</tr>
<tr>
<td>TINYINT</td>
<td>NUMBER(3, 0)</td>
<td></td>
</tr>
</tbody>
</table>
Deploying Oracle Extractors

<table>
<thead>
<tr>
<th>MySQL Datatype</th>
<th>Oracle Datatype</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMALLINT</td>
<td>NUMBER(5, 0)</td>
<td></td>
</tr>
<tr>
<td>MEDIUMINT</td>
<td>NUMBER(7, 0)</td>
<td></td>
</tr>
<tr>
<td>DECIMAL(x,y)</td>
<td>NUMBER(x, y)</td>
<td></td>
</tr>
<tr>
<td>FLOAT</td>
<td>FLOAT</td>
<td></td>
</tr>
<tr>
<td>CHAR(n)</td>
<td>CHAR(n)</td>
<td></td>
</tr>
<tr>
<td>VARCHAR(n)</td>
<td>VARCHAR2(n)</td>
<td>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.</td>
</tr>
<tr>
<td>DATE</td>
<td>DATE</td>
<td></td>
</tr>
<tr>
<td>DATETIME</td>
<td>DATE</td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>DATE</td>
<td></td>
</tr>
<tr>
<td>TEXT</td>
<td>CLOB</td>
<td>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(N) can be used on Oracle. This may be faster than using CLOB.</td>
</tr>
<tr>
<td>BLOB</td>
<td>BLOB</td>
<td></td>
</tr>
<tr>
<td>ENUM(...)</td>
<td>VARCHAR(255)</td>
<td>Use the enuntostring filter</td>
</tr>
<tr>
<td>SET(...)</td>
<td>VARCHAR(255)</td>
<td>Use the settostring filter</td>
</tr>
</tbody>
</table>

When replicating between Oracle and other database types, the ddlscan command can be used to generate DDL appropriate for the supported data types in the target database. For example, 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.

When replicating from Oracle to MySQL or Oracle, there are limitations on the data types that can be replicated due to the nature of the CDC, whether you are using Asynchronous or Synchronous CDC for replication. The details of data types not supported by each mechanism are detailed in Table 4.2, “Data Type Differences when Replicating from Oracle to MySQL or Oracle”.

### Table 4.2. Data Type Differences when Replicating from Oracle to MySQL or Oracle

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Asynchronous CDC [Oracle EE Only]</th>
<th>Synchronous CDC [Oracle SE and EE]</th>
</tr>
</thead>
<tbody>
<tr>
<td>BFILE</td>
<td>Not Supported</td>
<td>Not Supported</td>
</tr>
<tr>
<td>LONG</td>
<td>Not Supported</td>
<td>Not Supported</td>
</tr>
<tr>
<td>ROWID</td>
<td>Not Supported</td>
<td>Not Supported</td>
</tr>
<tr>
<td>UROWID</td>
<td>Not Supported</td>
<td>Not Supported</td>
</tr>
<tr>
<td>BLOB</td>
<td>Not Supported</td>
<td>Not Supported</td>
</tr>
<tr>
<td>CLOB</td>
<td>Not Supported</td>
<td>Not Supported</td>
</tr>
<tr>
<td>NCLOB</td>
<td>Not Supported</td>
<td>Not Supported</td>
</tr>
<tr>
<td>All Object Types</td>
<td>Not Supported</td>
<td>Not Supported</td>
</tr>
</tbody>
</table>

**Note**

More information on Oracle CDC can be found within the Oracle documentation.

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 Clustering 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 Clustering includes a tool called `ddlscan` which will read table definitions from MySQL and create appropriate Oracle table definitions to use during replication. For more information on using this tool in a MySQL to Oracle deployment, see Section 5.7.1.3, “Create the Destination Schema”.

For reference information on the `ddlscan` tool, see Section 8.6, “The `ddlscan` Command”.

### 4.1.3. Creating an Oracle to MySQL Deployment

<table>
<thead>
<tr>
<th>Replication Operation Support</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Statements Replicated</td>
<td>No</td>
</tr>
<tr>
<td>Rows Replicated</td>
<td>Yes</td>
</tr>
<tr>
<td>Schema Replicated</td>
<td>No</td>
</tr>
<tr>
<td><code>ddlscan</code> Supported</td>
<td>Yes</td>
</tr>
</tbody>
</table>

The Oracle extractor enables information to be extracted from an Oracle database, generating row-based information that can be replicated to other replication services, including MySQL. The transactions are extracted by Oracle by capturing the change events and writing them to change tables; Tungsten Replicator extracts the information from the change tables and uses this to generate the row-changed data that is then written to the THL and applied to the destination.

Replication from Oracle has the following parameters:

• Data is replicated using row-based replication; data is extracted by row from the source Oracle database and applied by row to the target MySQL database.

• DDL is not replicated; schemas and tables must be created on the target database before replication starts.

• Tungsten Replicator relies on two different users within Oracle configuration the configuration, both are created automatically during the CDC configuration:
  1. **Publisher** — the user designated to issue the CDC commands and generates and is responsible for the CDC table data.
  2. **Subscriber** — the user that reads the CDC change table data for translation into THL.

• The slave replicator (applier), writes information into the target MySQL database using a standard JDBC connection.

The basic process for creating an Oracle to MySQL replication is as follows:

1. **Configure the Oracle database**, including configuring users and CDC configuration.
2. **Configure the MySQL database**, including creating tables and schemas.
3. **Extract the schema from MySQL and translate it to Oracle DDL**.
4. **Install the Master replicator** to extract information from the Oracle database.
5. **Install the Slave replicator** to read data from the master database and apply it to MySQL.

### 4.1.3.1. Configuring the Oracle Environment

The primary stage in configuring Oracle to MySQL replication is to configure the Oracle environment and databases ready for use as a data source by the Tungsten Replicator. A script, `setupCDC.sh` automates some of the processes behind the initial configuration and is responsible for creating the required Change Data Capture tables that will be used to capture the data change information.

Before running `setupCDC.sh`, the following steps must be completed.

• Ensure that Oracle is configured to accept dates in `YYYY-MM-DD` format used by Tungsten Replicator.
Deploying Oracle Extractors

sqlplus sys/oracle as sysdba
SQL> ALTER SYSTEM SET NLS_DATE_FORMAT='YYYY-MM-DD' SCOPE=SPFILE;

Then restart the database for the change to take effect:

sqlplus sys/oracle as sysdba
SQL> shutdown immediate
SQL> startup

• Create the source user and schema if it does not already exist.

Once these steps have been completed, a configuration file must be created that defines the CDC configuration. For more information on the options for setupCDC.conf, see Section 8.14, “The setupCDC.sh Command”.

A sample configuration file is provided in tungsten-replicator/exTRACTORS/oracle-cdc/setupCDC.conf within the distribution directory.

To configure the CDC configuration:

1. For example, the following configuration would setup CDC for replication from the sales schema (comment lines have been removed for clarity):

```
   service=SALES
   sys_user=sys
   sys_pass=oracle
   enable_archive_log=0
   export source_user=sales
   pub_tablespace=0
   pub_user=$(source_user)_pub
   pub_password=password
   tungsten_user=tungsten
   tungsten_pwd=password
   delete_publisher=0
   delete_subscriber=0
   cdc_type=HOTLOG_SOURCE
   specific_tables=0
   specific_path=
```

2. Before running setupCDC.sh, and if you have set the pub_tablespace variable to 1, you must create the tablespace that will be used to hold the CDC data. This needs to be created only once:

```
   SQL> CREATE TABLESPACE "SALES_PUB" DATAFILE '/oracle/SALES_PUB' SIZE 10485760 AUTOEXTEND ON NEXT 1048576 MAXSIZE 32767M NOLOGGING ONLINE PERMANENT BLOCKSIZE 8192 EXTENT MANAGEMENT LOCAL AUTOALLOCATE DEFAULT NOCOMPRESS SEGMENT SPACE MANAGEMENT AUTO;
```

The above SQL statement is all one statement. The tablespace name and data file locations should be modified according to the pub_user values used in the configuration file. Note that the directory specified for the data file must exist, and must be writable by Oracle.

3. Once the configuration file has been created, run setupCDC.sh with the configuration file (it defaults to setupCDC.conf). The command must be executed within the tungsten-replicator/scripts within the distribution (or installation) directory, as it relies on SQL scripts in that directory to operate:

```
   shell> cd tungsten-replicator-5.4.1-41/tungsten-replicator/exTRACTORS/oracle-cdc
   shell> ./setupCDC.sh custom.conf
```

The above output shows the steps taken by the setupCDC.sh script. Each step is described in detail in the accompanying documentation.
The script will report the current CDC archive log position where extraction will start.

If there are error, the problem with the script and setup will be reported. The problem should be corrected, and the script executed again until it completes successfully.

Once the CDC configuration has completed, the Tungsten Replicator is ready to be installed.

4.1.3.2. Creating the MySQL Environment

The MySQL side can be a standard MySQL installation, including the Appendix B, Prerequisites required for all Tungsten Replicator services.

In particular:

- The `tungsten` user, or configured datasource user, must have been created to enables writes to MySQL, and been granted suitable permissions.

- Information from the Oracle server is replicated in row-based format which implies additional disk space overhead, so you must ensure that you have enough disk space for the THL files.

When writing the row data into MySQL, Tungsten Replicator supports two different modes of operation:

- Write row-columns in order — the default mode, columns are written to MySQL in the same order in which they are extracted from the Oracle database. This allows for differences in the table and column names in the target database, while still replicating the same information. This can be useful if there are reserved words or other differences between the two environments.

- Write using column-names — this enables the column orders to be different, but the column names to be used to apply data. This can be particularly useful if only a selection of columns are being extracted from Oracle and these selected columns are being written into MySQL. To enable this option, the following setting must be applied to the `tpm` installation command used:

```
--property=replicator.applier.dbms.getColumnMetadataFromDB=false
```

4.1.3.3. Creating the Destination Schema

On the host which has been already configured as the master, use `ddlscan` to extract the DDL for Oracle:

```
shell> cd tungsten-replicator-5.4.1-41
shell> ./bin/ddlscan -user tungsten -url 'jdbc:oracle:thin:@//host1:1521/ORCL' \
    -pass password -db access_log -template ddl-oracle-mysql.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:oracle:thin:@//host1:1521/ORCL' \
    -pass password -template ddl-oracle-mysql.vm -db access_log > access_log.ddl
```

The generate DDL should be checked, particularly for comments with `ERROR` in them, as they indicate unsupported types. If you are happy with the output, it can be executed against your target Oracle database:

```
shell> cat access_log.ddl | mysql
```

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 MySQL to Oracle.

In addition, the process should also be performed for the master `tungsten_alpha` database to ensure that the table definitions are migrated correctly.

4.1.3.4. Creating the Master Replicator

The master replicator reads information from the CDC tables and converts that information into THL, which can then be replicated to other Tungsten Replicator installations. The basic operation is to create an installation using `tpm`, using the connection information provided when executing the CDC configuration, including the subscriber and CDC type.

1. Unpack the Tungsten Replicator distribution in staging directory:

```
shell> tar zxf tungsten-replicator-5.4.tar.gz
```

2. Change into the staging directory:
3. Obtain a copy of the Oracle JDBC driver and copy it into the `tungsten-replicator/lib` directory:

```
shell> cp ojdbc6.jar ./tungsten-replicator/lib/
```

4. Execute the following command to install the Oracle extractor:

```
shell> ./tools/tpm install SALES \
    --datasource-oracle-service=ORCL \
    --install-directory=/opt/continuent \
    --master=host1 \
    --members=host1 \
    --property=replicator.extractor.dbms.transaction_frag_size=10 \
    --property=replicator.global.extract.db.password=password \
    --property=replicator.global.extract.db.user=tungsten \
    --property=replicator.password=password \
    --property=replicator.port=1521 \
    --property=replicator.user=SALES_PUB \
    --role=master \
    --start-and-report=true \
    --svc-table-engine=CDCASYNC
```

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:

- **tpm install SALES**
  
  Install the service, using `SALES` as the service name, using `tpm install`. This must match the service name given when running `setupCDC.sh`.

- **--datasource-oracle-service=ORCL** [374]
  
  Specify the Oracle service name, as configured for the database to which you want to read data. For older Oracle installations that use the SID format, use the `--datasource-oracle-sid=ORCL` option to `tpm`.

- **--datasource-type=oracle** [375]
  
  Defines the datasource type that will be read from, in this case, Oracle.

- **--install-directory=/opt/continuent** [382]
  
  The installation directory for Tungsten Clustering.

- **--master=host1** [386]
  
  The hostname of the master.

- **--members=host1** [386]
  
  The list of members for this service.

- **--property=replicator.extractor.dbms.transaction_frag_size=10** [334]
  
  Define the fragment size, or number of transactions that will be queued before extraction.

- **--property=replicator.global.extract.db.password=password** [334]
  
  The password of the subscriber user configured within `setupCDC.sh`.

- **--property=replicator.global.extract.db.user=tungsten** [334]
  
  The username of the subscriber user configured within `setupCDC.sh`.

- **--replicator-host=host1** [396]
  
  The hostname of the replicator.

- **--replicator-password=password** [396]
  
  The password of the CDC publisher, as defined within the `setupCDC.sh`.

- **--replicator-port=1521** [396]
Deploying Oracle Extractors

The port used to read information from the Oracle server. The default port is port 1521.

- `--replication-user=SALES_PUB`[397]

  The name of the CDC publisher, as defined within the `setupCDC.sh`.

- `--role=master`[397]

  The role of the replicator, the replicator will be installed as a master extractor.

- `--start-and-report=true`[399]

  Start the replicator and report the status.

- `--svc-table-engine=CDCASYNC`[401]

  The type of CDC extraction that is taking place, the same value as used the `setupCDC.conf` is supported, i.e. using `CDCSYNC` or `CDCASYNC` accordingly.

<table>
<thead>
<tr>
<th><code>setupCDC.conf</code> Setting</th>
<th><code>--svc-table-engine</code> Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SYNC_SOURCE</code></td>
<td><code>CDCSYNC</code></td>
</tr>
<tr>
<td><code>HOTLOG_SOURCE</code></td>
<td><code>CDCASYNC</code></td>
</tr>
</tbody>
</table>

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

Once the replicator has been installed, the current status of the replicator can be checked using `trepctl status`:

```
shell> trepctl status
Processing status command...
NAME        VALUE
----        -----...
appliedLastEventId   : ora:16626156
appliedLastSeqno     : 67
appliedLatency       : 37.51
autoRecoveryEnabled  : false
autoRecoveryTotal    : 0
channels             : 1
clusterName          : SALES
currentTimeMillis    : 1410430937700
dataServerHost       : tr-fromoracle1
extensions           : 
host                 : tr-fromoracle1
latestEpochNumber    : 67
masterConnectUri     : thl://localhost:
masterListenUri      : thl://tr-fromoracle1:2112/
maximumStoredSeqNo   : 67
minimumStoredSeqNo   : 67
offlineRequests      : NONE
pendingError         : NONE
pendingErrorCode     : NONE
pendingErrorEventId  : NONE
pendingErrorSeqno    : -1
pendingExceptionMessage: NONE
pipelineSource       : UNKNOWN
relativeLatency      : 38.699
resourcePrecedence   : 99
rmiPort              : 10000
role                 : master
serviceType          : SALES
siteName             : default
sourceId             : tr-fromoracle1
state                : ONLINE
uptimeSeconds         : 37.782
transitionalTo       : 
useSslConnection     : false
version              : Tungsten Replicator 5.4.1 build 41
Finished status command...
```
4.1.3.5. Creating the Slave Replicator

The MySQL slave applier is a simple applier that writes the data from the Oracle replicator into MySQL. The replicator can be installed using `tpm`. The base configuration can be achieved with just a few options, but for convenience, additional filters are employed to change the case of the schema (Oracle schemas are normally in uppercase, MySQL in lowercase), and to rename the Tungsten specific tables so that they match the required service name. For example, within Oracle, the Tungsten tables are stored within the pub user tablespace (i.e. `SALES_PUB` for the `SALES` user), but in a MySQL deployment these tables are stored within a database named after the service (i.e. `tungsten_alpha`).

1. Unpack the Tungsten Replicator distribution in staging directory:

   ```shell
tar zxf tungsten-replicator-5.4.tar.gz
   ```

2. Change into the staging directory:

   ```shell
cd tungsten-replicator-5.4
   ```

3. Obtain a copy of the Oracle JDBC driver and copy it into the `tungsten-replicator/lib` directory:

   ```shell
cp ojdbc6.jar ./tungsten-replicator/lib/
   ```

4. These requirements lead to a `tpm` configuration as follows:

   ```shell
   ./tools/tpm install alpha \
   --install-directory=/opt/continuent \
   --master=host1 \
   --members=host2 \
   --datasource-password=password \
   --datasource-user=tungsten \
   --svc-applier-filters=CDC,dbtransform,optimizeupdates \
   --property=replicator.filter.CDC.from=SALES_PUB.HEARTBEAT \
   --property=replicator.filter.CDC.to=tungsten_alpha.heartbeat \
   --property=replicator.filter.dbtransform.from_regex1=DEMO \
   --property=replicator.filter.dbtransform.to_regex1=demo \
   --skip-validation-check=InstallerMasterSlaveCheck \
   --start-and-report
   ```

Once the service has started, the status can be checked and monitored by using the `trepctl` command.

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.

- `--members=host2` [386]
  Specifies the members of the cluster. In this case, the only member is the host into which we are deploying the slave replicator service.

- `--master=host1` [386]
  Specify the name of the master replicator that will provide the THL data to be replicated, in this case, the Oracle server configured with the CDC service.

- `--datasource-user=tungsten_alpha` [397]
  The name of the user created within Oracle to be used for writing data into the Oracle tables.

- `--datasource-password=password` [396]
  The password to be used by the Oracle user when writing data.

- `--install-directory=/opt/continuent` [382]
  The directory where Tungsten Replicator will be installed.

- `--svc-applier-filters=dropstatementdata` [399]
  Enables a filter that will ensure that statement information is dropped. When executing statement data that was written from MySQL, those statements cannot be executed on Oracle, so the statements are filtered out using the `dropstatementdata` filter.

- `--skip-validation-check=InstallerMasterSlaveCheck` [335]
  Skip validation for the MySQL master/slave operation, since that it is irrelevant in a MySQL/Oracle deployment.

- `--start-and-report` [399]
Deploying Oracle Extractors

Start the service and report the status.

- `--svc-applier-filters=CDC,dbtransform,optimizupdates` [399]

Enable a number of filters to improve the replication:

- `cdcmetadata` filter renames tables from a CDC deployment to a corresponding MySQL table.
- `dbtransform` enables regex-based table renaming.
- `optimizupdates` alters table updates to be more efficient by removing values from the ROW update statement that have not changed.
- `--property=replicator.filter.CDC.from=SALES_PUB.HEARTBEAT` [334]

Specifies the table from which the table names will be converted for CDC metadata.

- `--property=replicator.filter.CDC.to=tungsten_alpha.heartbeat` [334]

Defines the target name for the CDC metadata rename.

- `--property=replicator.filter.dbtransform.from_regex1=DEMO` [334]

Specifies the regex pattern match, in this case, the name of the database in uppercase format, as it will be extracted from Oracle.

- `--property=replicator.filter.dbtransform.to_regex1=demo` [334]

The target regex format for matching tables. In this case, uppercase names (DEMO) will be renamed to demo.

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

Once installed, check the replicator status using `trepctl status`:

```
shell> trepctl status
Processing status command...
NAME         VALUE
......
appliedLastEventId : ora:16626156
appliedLastSeqno   : 67
appliedLatency     : 314.359
channels           : 1
clusterName        : alpha
currentEventId     : NONE
currentTimeMillis  : 1410431215649
dataServerHost     : tr-fromoracle2
extensions         :
host               : tr-fromoracle2
latestEpochNumber  : 67
masterConnectUri   : thl://tr-fromoracle1:2112/
masterListenUri    : thl://tr-fromoracle2:2112/
maximumStoredSeqNo : 67
minimumStoredSeqNo : 67
offlineRequests    :
pendingError       :
pendingErrorCode   :
pendingErrorEventId :
pendingErrorSeqno  :
pendingExceptionMessage:
pipelineSource     : thl://tr-fromoracle1:2112/
relativeLatency    : 316.649
resourcePrecedence : 99
rmiPort            : 10000
role               : slave
seqnoType          : java.lang.Long
serviceName        : alpha
serviceType        : local
simpleServiceName  : alpha
siteName           : default
sourceId           : tr-fromoracle2
state              : ONLINE
timeInStateSeconds : 2.343
transitioningTo    :
uptimeSeconds      : 74327.712
useSSLConnection   : False
version            : Tungsten Replicator 5.4.1 build 41
Finished status command...
```
An Oracle to Oracle deployment replicates data between two Oracle servers, either for scaling or Data Recovery (DR) support. Enabling the Oracle to Oracle replication consists of the following parameters during replication:

- Data is replicated using row-based replication; data is extracted by row from the source Oracle database and applied by row to the target Oracle database.
- DDL is not replicated; schemas and tables must be created on the target database before replication starts.
- Tungsten Replicator relies on two different users within Oracle configuration the configuration, both are created automatically during the CDC configuration:
  - Publisher — the user designated to issue the CDC commands and generates and is responsible for the CDC table data.
  - Subscriber — the user that reads the CDC change table data for translation into THL.
- The slave replicator [applier], writes information into the target Oracle database using a standard JDBC connection.

The basic process for creating an Oracle to Oracle deployment is:

- Configure the source (master) Oracle database, including configuring users and CDC configuration to extract the data.
- Prepare the target Oracle database. Users must be created, and the DDL from the source Oracle server applied to the target database before replication begins.
- Create the schema on the target Oracle database.
- Install the Master replicator to extract information from the Oracle database.
- Install the Slave replicator to read data from the master replicator and apply it to Oracle.

### 4.1.4.1. Setting up the Source Oracle Environment

The primary stage in configuring Oracle to MySQL replication is to configure the Oracle environment and databases ready for use as a data source by the Tungsten Replicator. A script, `setupCDC.sh`, automates some of the processes behind the initial configuration and is responsible for creating the required Change Data Capture tables that will be used to capture the data change information.

Before running `setupCDC.sh`, the following steps must be completed.

- Ensure that Oracle is configured to accept dates in `YYYY-MM-DD` format used by Tungsten Replicator:

  ```sql
  sqlplus sys/oracle as sysdba
  SQL> ALTER SYSTEM SET NLS_DATE_FORMAT='YYYY-MM-DD' SCOPE=SPFILE;
  SQL>
  SQL> shutdown immediate
  SQL> startup
  ```

- Create the source user and schema if it does not already exist.

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

Once these steps have been completed, a configuration file must be created that defines the CDC configuration.

**Table 4.3. `setupCDC.conf` Configuration File Parameters**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Sample Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>service</td>
<td></td>
<td>The name of the service that will be used to process these events. It should match the name of the schema from which data is being read. The name should also match the name of the service that will be created using Tungsten Replicator to extract events from Oracle.</td>
</tr>
</tbody>
</table>
Deploying Oracle Extractors

<table>
<thead>
<tr>
<th>Variable</th>
<th>Sample Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys_user [270]</td>
<td>SYSDBA</td>
<td>The name of the SYSDBA user configured. The default (if not specified) is SYSDBA.</td>
</tr>
<tr>
<td>sys_pass</td>
<td></td>
<td>The password of the SYSDBA user; you will be prompted for this information if it has not been added.</td>
</tr>
<tr>
<td>enable_archivelog</td>
<td>0</td>
<td>If set to true, the Oracle instance will be configured to enable archive logging (required by Oracle CDC), and then the Oracle instance will be restarted.</td>
</tr>
<tr>
<td>source_user</td>
<td></td>
<td>The name of the source schema user that will be used to identify the tables used to build the publish tables. This user is created by the setupCDC.sh script.</td>
</tr>
<tr>
<td>pub_tablespace</td>
<td>0</td>
<td>By default, the system tablespace is used for holding the publisher tables. Using the system tablespace should only be used during testing, as the tablespace is typically not large enough to hold the required change data. If set to 1, use the created tablespace (matching the value of pub_user) which is assumed to be large enough to hold the change information.</td>
</tr>
<tr>
<td>pub_user</td>
<td></td>
<td>The publisher user that will be created to publish the CDC views.</td>
</tr>
<tr>
<td>pub_password</td>
<td></td>
<td>The publisher password that will be used when the publisher user is created.</td>
</tr>
<tr>
<td>tungsten_user</td>
<td>tungsten</td>
<td>The subscriber user that will be created to access the CDC views. This will be used as the datasource username within the Tungsten Replicator configuration.</td>
</tr>
<tr>
<td>tungsten_pwd</td>
<td>password</td>
<td>The subscriber password that will be created to access the CDC. This will be used as the datasource username within the Tungsten Replicator configuration. views.</td>
</tr>
<tr>
<td>delete_publisher</td>
<td></td>
<td>If set to 1, the publisher user will be deleted before being recreated.</td>
</tr>
<tr>
<td>delete_subscriber</td>
<td></td>
<td>If set to 1, the subscriber user will be deleted before being recreated.</td>
</tr>
<tr>
<td>cdc_type [268]</td>
<td>SYNC_SOURCE</td>
<td>Specifies the CDC extraction type to be deployed. Using SYNC_SOURCE uses synchronous capture; HOTLOG_SOURCE uses asynchronous capture.</td>
</tr>
<tr>
<td>specific_tables</td>
<td></td>
<td>If set to 1, limits the replication to only use the tables listed in a tungsten.tables file. If set to 0, no file is used and all tables are included.</td>
</tr>
<tr>
<td>specific_path</td>
<td></td>
<td>The path of the tungsten.tables file. When using Oracle RAC, the location of the tungsten.tables file must be in a shared location accessible by Oracle RAC. If not specified, the current directory is used.</td>
</tr>
</tbody>
</table>

A sample configuration file is provided in tungsten-replicator/scripts/setupCDC.conf within the distribution directory.

To configure the CDC configuration:

1. For example, the following configuration would setup CDC for replication from the sales schema (comment lines have been removed for clarity):

```bash
service=SALES
sys_user=sys
sys_pass=oracle
enable_archive_log=0
export source_user=sales
pub_tablespace=0
pub_user=${source_user}_pub
pub_password=password
tungsten_user=tungsten
tungsten_pwd=password
delete_publisher=0
delete_subscriber=0
cdc_type=HOTLOG_SOURCE
specific_tables=0
specific_path=
```

2. Before running setupCDC.sh, and if you have set the `pub_tablespace` variable to 1, you must create the tablespace that will be used to hold the CDC data. This needs to be created only once:

```bash
bash
shell> sqlplus sys/oracle as sysdba
SQL> CREATE TABLESPACE "SALES_PUB" DATAFILE '/oracle/SALES_PUB' SIZE 10485760 AUTOEXTEND ON NEXT
```
Deploying Oracle Extractors

```
1048576 MAXSIZE 32767M NOLOGGING ONLINE PERMANENT BLOCKSIZE 8192 EXTENT MANAGEMENT LOCAL AUTOALLOCATE
DEFAULT NOCOMPRESS SEGMENT SPACE MANAGEMENT AUTO;
```

The above SQL statement is all one statement. The tablespace name and data file locations should be modified according to the `pub_user` values used in the configuration file. Note that the directory specified for the data file must exist, and must be writable by Oracle.

3. Once the configuration file has been created, run `setupCDC.sh` with the configuration file [It defaults to `setupCDC.conf`]. The command must be executed within the `tungsten-replicator/scripts` within the distribution [or installation] directory, as it relies on SQL scripts in that directory to operate:

```
shell> cd tungsten-replicator-5.4.1-41/tungsten-replicator/scripts
shell> ./setupCDC.sh custom.conf
Using configuration custom.conf
Configuring CDC for service ‘SALES’ for Oracle 11. Change Set is ‘TUNGSTEN_CS_SALES’
Removing old CDC installation if any (SYSDBA)
Done.
Setup tungsten_load (SYSDBA)
Done.
Creating publisher/subscriber and preparing table instantiation (SYSDBA)
Done.
Setting up HOTLOG_SOURCE (SALES_PUB)
Oracle version : 11.2.0.2.0
Setting up Asynchronous Data Capture TUNGSTEN_CS_SALES
Processing SALES.SAMPLE -> ‘CT_SAMPLE’ : OK
Enabling change set : TUNGSTEN_CS_SALES
Dropping view TUNGSTEN_PUBLISHED_COLUMNS
Dropping view TUNGSTEN_SOURCE_TABLES
PL/SQL procedure successfully completed.
Done.
adding synonym if needed (tungsten)
Done.
Cleaning up (SYSDBA)
Done.
Capture started at position 16610205
```

The script will report the current CDC archive log position where extraction will start.

If there are error, the problem with the script and setup will be reported. The problem should be corrected, and the script executed again until it completes successfully.

4.1.4.2. Setting up the Target Oracle Environment

Before starting 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 Clustering 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 user. 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 source schema 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 ALL PRIVILEGES TO tungsten_alpha;
```

The user/password combination selected will be required when configuring the slave replication service.

4.1.4.3. Creating the Destination Schema

When replicating from Oracle to Oracle, the schema of the two tables should match, or at least be compatible, if you are filtering or renaming tables. If the schema on the source Oracle server is available, it should be used to generate the schema on the destination server.

Tables should be created on the slave with the following caveats:
• Drop triggers from all tables. Triggers are not automatically disabled by Tungsten Replicator, and leaving them enabled may create data drift, duplicate keys and other errors. Triggers can be disabled on a table using:

```sql
SQL> ALTER TABLE sales DISABLE ALL TRIGGERS
```

• Remove foreign key constraints. Because data is replicated based on the window of changes provided for each CDC block, and the order of the individual operations may not be identical. This can lead to foreign key constraints failing, even though the source database updates were processed correctly.

If the schema is not separately available, the schema information can be extracted within `sqlplus`, either by display the table definition using the `DESC` command:

```sql
SQL> desc SALES.sample;
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Null?</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>NOT NULL</td>
<td>NUMBER(38)</td>
</tr>
<tr>
<td>MSG</td>
<td>CHAR(80)</td>
<td></td>
</tr>
</tbody>
</table>

Or by extracting the information from the database metadata:

```sql
SQL> select dbms_metadata.get_ddl('TABLE','SAMPLE','SALES') from dual;
```

DBMS_METADATA.GET_DDL('TABLE','SAMPLE','SALES')

| CREATE TABLE "SALES"."SAMPLE" |
| ( "ID" NUMBER(*,0), |
| "MSG" CHAR(80), |
| PRIMARY KEY ("ID") |
| USING INDEX PCTFREE 10 INITrans 2 MAXTRANS 255 COMPUTE STATISTICS NOLOGGING |
| STORAGE(INITIAL 65536 NEXT 184576 MINEXTENTS 1 MAXEXTENTS 2147483645 |
| PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE DE |
| FAULT CELL_FLASH_CACHE DEFAULT) |
| TABLESPACE "SALES_PUB" ENABLE, |
| SUPPLEMENTAL LOG DATA (PRIMARY KEY) COLUMNS, |
| DBMS_METADATA.GET_DDL('TABLE','SAMPLE','SALES')
```

Note that the information may be truncated due to the configuration only displaying a subset of the generated `LONG` datatype used to display the information. The command:

```sql
SQL> set long 10000;
```

Will increase the displayed length to 10,000 characters.

4.1.4.4. Installing the Master Replicator

The master replicator reads information from the CDC tables and converts that information into THL which can then be replicated to other Tungsten Replicator installations. The basic operation is to create an installation using `tpm`, using the connection information provided when executing the CDC configuration, including the subscriber and CDC type.

1. Unpack the Tungsten Replicator distribution in staging directory:

   ```sh
tenhr> tar zxf tungsten-replicator-5.4.tar.gz
   ```

2. Change into the staging directory:

   ```sh
tenhr> cd tungsten-replicator-5.4
   ```

3. Obtain a copy of the Oracle JDBC driver and copy it into the `tungsten-replicator/lib` directory:

   ```sh
tenhr> cp ojdbc6.jar ./tungsten-replicator/lib/
   ```

4. Execute the `install` command for the master replicator:

   ```sh
   tenhr> ./tools/tpm install SALES --datasource-oracle-service=ORCL --install-directory=/opt/continuent --master=host1 --members=host1 --property=replicator.extractor.dbms.transaction_frag_size=10
   ```
Deploying Oracle Extractors

```bash
--property=replicator.global.extract.db.password=password \
--property=replicator.global.extract.db.user=tungsten \
--replication-host=host1 \
--replication-password=password \
--replication-port=1521 \
--replication-user=SALES_PUB \
--role=master \
--start-and-report=true \
--svc-table-engine=CDCASYNC
```

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.

- tpm install SALES
  
  Install the service, using `SALES` as the service name, using `tpm install`. This must match the service name given when running `setupCDC.sh`

- `--datasource-oracle-service=ORCL`
  
  Specify the Oracle service name, as configured for the database to which you want to read data. For older Oracle installations that use the SID format, use the `--datasource-oracle-sid=ORCL` option to `tpm`.

- `--datasource-type=oracle`
  
  Defines the datasource type that will be read from, in this case, Oracle.

- `--install-directory=/opt/continuent`
  
  The installation directory for Tungsten Clustering.

- `--master=host1`
  
  The hostname of the master.

- `--members=host1`
  
  The list of members for this service.

- `--property=replicator.extractor.dbms.transaction_frag_size=10`
  
  Define the fragment size, or number of transactions that will be queued before extraction.

- `--property=replicator.global.extract.db.password=password`
  
  The password of the subscriber user configured within `setupCDC.sh`

- `--property=replicator.global.extract.db.user=tungsten`
  
  The username of the subscriber user configured within `setupCDC.sh`

- `--replication-host=host1`
  
  The hostname of the replicator.

- `--replication-password=password`
  
  The password of the CDC publisher, as defined within the `setupCDC.sh`

- `--replication-port=1521`
  
  The port used to read information from the Oracle server. The default port is port 1521.

- `--replication-user=SALES_PUB`
  
  The name of the CDC publisher, as defined within the `setupCDC.sh`

- `--role=master`
  
  The role of the replicator, the replicator will be installed as a master extractor.

- `--start-and-report=true`
  
  77
Start the replicator and report the status.

- --svc-table-engine=CDCASYNC [401]

The type of CDC extraction that is taking place is the same value as used in the setupCDC.conf is supported, i.e. using CDCSYNC or CDCASYNC accordingly.

<table>
<thead>
<tr>
<th>setupCDC.conf Setting</th>
<th>--svc-table-engine [401] Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNC_SOURCE</td>
<td>CDCSYNC</td>
</tr>
<tr>
<td>HOTLOG_SOURCE</td>
<td>CDCASYNC</td>
</tr>
</tbody>
</table>

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

Once the replicator has been installed, the current status of the replicator can be checked using `trepctl status`:

```shell
shell> trepctl status
Processing status command...
NAME       VALUE
----       -----             
appliedLastEventId : ora:16626156
appliedLastSeqno  : 67
appliedLatency   : 37.51
autoRecoveryEnabled : false
autoRecoveryYtd : 0
channels : 1
clusternamem : SALES
currentEventId : NONE
currentTimeMillis : 1410430937700
dataServerHost : tr-fromoracle1
extension : 
host : tr-fromoracle1
latestEpochNumber : 67
masterConnectUri : thl://localhost:/
masterListenUri : thl://tr-fromoracle1:2112/
maximumStoriedSeqNo : 67
minimumStoriedSeqNo : 67
offlineRequests : NONE
pendingError : NONE
pendingErrorEventId : NONE
pendingErrorSeqno : -1
pendingExceptionMessage : NONE
pipelineSource : UNKNOWN
relativeLatency : 38.699
resourcePrecedence : 99
rmiPort : 10000
role : master
seqnoType : java.lang.Long
serviceName : SALES
serviceType : local
simpleServiceName : SALES
siteName : default
sourceId : tr-fromoracle1
state : ONLINE
timeInStateSeconds : 37.782
transitioningTo : 
uptimeSeconds : 102.545
useSSLConnection : false
version : Tungsten Replicator 5.4.1 build 41
Finished status command...
```

4.1.4.5. Installing the Slave Replicator

The slave replicator will read the THL from the remote master and apply it into Oracle using a standard JDBC connection. The slave replicator needs to know the master hostname, and the datasource type.

1. Unpack the Tungsten Replicator distribution in staging directory:

```shell
shell> tar zxf tungsten-replicator-5.4.tar.gz
```

2. Change into the staging directory:

```shell
shell> cd tungsten-replicator-5.4
```

3. Obtain a copy of the Oracle JDBC driver and copy it into the tungsten-replicator/lib directory:

```shell
shell> cp ojdbc6.jar ./tungsten-replicator/lib/
```
Deploying Oracle Extractors

4. Install the Slave replicator to read data from the master database and apply it to Oracle:

```
shell> /tools/tpm install SALES \
  --members=host2 \
  --master=host1 \
  --datasource-type=oracle \
  --datasource-oracle-service=ORCL \
  --datasource-user=tungsten \
  --datasource-password=password \
  --install-directory=/opt/continuent \
  --svc-applier-filters=dropstatementdata \
  --skip-validation-check=InstallerMasterSlaveCheck \
  --start-and-report
```

Once the service has started, the status can be checked and monitored by using the `trepctl` command.

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.

- `--members=host2` [386]
  Specifies the members of the cluster. In this case, the only member is the host into which we are deploying the slave replicator service.

- `--master=host1` [386]
  Specify the name of the master replicator that will provide the THL data to be replicated.

- `--datasource-type=oracle` [375]
  Specify the datasource type, in this case Oracle. This configures the replicator to use the Oracle JDBC driver, semantics, and connect to the Oracle database to manage the replication service.

- `--datasource-oracle-service=ORCL` [374]
  The name of the Oracle service within the Oracle database that the replicator will be writing data to. For older Oracle installations, where there is an explicit Oracle SID, use the `--datasource-oracle-sid` command-line option to `tpm`.

- `--datasource-user=tungsten_alpha` [397]
  The name of the user created within Oracle to be used for writing data into the Oracle tables.

- `--datasource-password=password` [396]
  The password to be used by the Oracle user when writing data.

- `--install-directory=/opt/continuent` [382]
  The directory where Tungsten Replicator will be installed.

- `--svc-applier-filters=dropstatementdata` [399]
  Enables a filter that will ensure that statement information is dropped. When executing statement data that was written from MySQL, those statements cannot be executed on Oracle, so the statements are filtered out using the `dropstatementdata` filter.

- `--skip-validation-check=InstallerMasterSlaveCheck` [335]
  Skip validation for the MySQL master/slave operation, since that it is irrelevant in a MySQL/Oracle deployment.

- `--start-and-report` [399]
  Start the service and report the status.

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 master replicator.

4.1.5. Deployment with Provisioning

You can setup the extractor from Oracle to automatically read and provision the slave database by using the Section 7.11, "Using the Parallel Extractor". The parallel extractor reads information from the source database schema in chunks and then feeds this information into the THL
Deploying Oracle Extractors

The data stream as row-based INSERT operations. When the slave connects, these are applied to the slave database as with a normal INSERT operations. The parallel extractor is particularly useful in heterogeneous environments such as Oracle to MySQL where the slave data does not already exist on the slave.

The basic provisioning process operates in two stages:

1. Provisioning data is extracted and inserted into the THL. One event is used to contain all of the data from a single table. If the table is too large to be contained in a single event, the data will be distributed over multiple events.

2. Once provisioning has finished, data is extracted from the CDC as normal and added to the THL

   Important
   The parallel extractor is not restart safe, and the process should not be interrupted.

This allows existing data to be extracted and processed through the replicator path, including filters within the applier. Once the initial data has been extracted, the change data to be applied.

To use the parallel extractor to provision data into the slave, the configuration must be performed as part of the installation process when configuring the master replicator.

To setup provisioning with parallel extractor:

1. Run `setupCDC.sh` to create the Oracle CDC infrastructure.

2. Install master Tungsten Replicator using `tpm`, but do not enable automatic starting (i.e. do not use the `--start [399]` or `--start-and-rep-port [399]` options).

3. On the slave database, create the destination tables for the schemas being replicated. This can be achieved either manually or by using `ddlscan` to create the required table definitions.

4. Install the slave replicator as normal; this can be a MySQL or Oracle destination.

5. On the master:
   a. Start the replicator in `OFFLINE [212]` mode using `replicator start offline`:

   ```shell`
   shell> replicator start offline
   ```

   b. Put the replicator into the `ONLINE [212]` state, using the `-provision` option:

   ```shell`
   shell> trepctl online -provision
   ```

   Alternatively, the system change number (SCN) identified when CDC capture is first enabled through `setupCDC.sh` can be used to provide a point-in-time provisioning. This can be useful if the data has previously been loaded and then CDC started by enabling provisioning from the start point. To use this method, identify the start position indicated by `setupCDC.sh`:

   ```shell`
   Capture started at position 40748375
   ```

   Then supply this to the `trepctl online -provision` command:

   ```shell`
   shell> trepctl online -provision 40748375
   ```

   During the provisioning process, the replicator will show the status `GOING-ONLINE:PROVISIONING [212]` until all of the data has been read from the existing database.

   The master will now start to read the information currently stored and feed this information through a separate pipeline into the THL.

6. On the slave, start the replicator, or put the replicator online. Statements from the master containing the provisioning information should be replicated into the slave.

   Important
   If the replicator is placed offline while the parallel extractor is still extracting data, the extraction process will continue to run and insert data until the extraction process has been completed.

Once the provisioned data has been inserted, replication will continue from the position where changes started to occur after the replicator was installed.

For more information on tuning the parallel extractor, see Section 7.11.1, “Advanced Configuration Parameters”.
4.1.6. Updating CDC after Schema Changes

If the schema for an existing CDC installation has changed, the CDC configuration must be updated to match the new schema configuration. If this step is not completed, then the correct information will not be extracted from the source tables into the CDC tables.

Schema changes should therefore be performed as follows:

1. Stop the replicator using `trepctl offline`:
   ```bash
   shell> trepctl offline
   ```

2. Change the schema definition within Oracle.

3. If multiple tables have been changed, update the `setupCDC.conf` file so that the `delete_publisher` variable is set to one. This will ensure that the publisher is dropped and recreated for the entire table set.

4. To update multiple tables, the entire setup process must be started again; run the `setupCDC.sh` command, using the original configuration file, but with the `delete_publisher` set to 1:
   ```bash
   shell> setupCDC.sh setupCDC.conf
   ```
   • If only one table has changed, or you are adding only a single table, this can be specified on the command-line:
   ```bash
   shell> updateCDC.sh setupCDC.conf samptable
   ```

5. Put the replicator back online with `trepctl online`:
   ```bash
   shell> trepctl online
   ```

To add a new table to an existing configuration:

1. Stop the replicator using `trepctl offline`:
   ```bash
   shell> trepctl offline
   ```

2. Update the configuration using `updateCDC.sh`, supplying the new table name.
   ```bash
   shell> updateCDC.sh setupCDC.conf newtable
   ```
   If you have used a specific tables file (i.e. with `specific_tables=1` in the configuration file, make sure that you add the table to the table file.

3. Put the replicator back online with `trepctl online`:
   ```bash
   shell> trepctl online
   ```

4.1.7. CDC Cleanup and Correction

In the event that the CDC tables have become corrupted, no longer work correctly, or where you have changed the tables, users or other details in your CDC configuration the CDC can be cleaned up. This deletes and unsubscribes the existing CDC configuration so that the `setupCDC.sh` script can be executed again with the updated values.

If `setupCDC.sh` returns an error that subscriptions already exist, this SQL file will also cleanup this configuration in preparation for running `setupCDC.sh` again.

To cleanup your existing configuration, an SQL script has been provided within the `tungsten-replicator/scripts` directory as `cleanup_cdc_tables.sql` for Oracle 11g, and `cleanup_cdc_tables-10.sql` for Oracle 10.

To execute, login to Oracle with `sqlplus` with SYSDBA credentials:

```bash
shell> sqlplus / as sysdba
SQL> @cleanup_cdc_tables.sql SALES_PUB TUNGSTEN_CS_SALES
```

Note

The changeset name used by every Tungsten Replicator CDC installation is prefixed with `TUNGSTEN_CS_`, followed by the service name configured in the CDC configuration file.

The name of the existing CDC publisher user and changeset should be specified to ensure that the right subscriptions are cleaned up.

Once completed, `setupCDC.sh` can be executed again. See Section 5.7.1.2, “Configure the Oracle database” for more information.
4.1.8. Tuning CDC Extraction

The frequency of extractions by the CDC extraction mechanism can be controlled by using the `maxSleepTime` parameter, which controls the maximum sleep time between data checks within the CDC tables. By default, the replicator checks for changes every second.

If there are no changes, the sleep time before the next query is increased by the `sleepAddition` until the value reaches, or is above, the `maxSleepTime` parameter. When changes are identified, the sleep time is reset back to 1 second. For example:

Increasing `maxSleepTime` sets the maximum sleep time and can help to reduce the overall redo log content generated, which in turn reduces the amount of disk space required to store and generate the log content. The value can be set during installation with `tpm` using:

```shell
tpm update alpha --property=replicator.extractor.dbms.maxSleepTime=32 ...
```

The minimum sleep time can be configured to set the minimum sleep time between checks of the CDC table using the `minSleepTime`. This reduces the size of the redo log, at the expense of lowering the frequency of updates from the CDC into THL. For example, setting a minimum of 16 seconds would change the sample interval shown in the previous table to:

<table>
<thead>
<tr>
<th>Sleep</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>16s</td>
<td>No data</td>
</tr>
<tr>
<td>32s</td>
<td>No data</td>
</tr>
<tr>
<td></td>
<td>Data found</td>
</tr>
<tr>
<td>16s</td>
<td>No data</td>
</tr>
<tr>
<td>32s</td>
<td>No data</td>
</tr>
<tr>
<td>32s</td>
<td>No data</td>
</tr>
<tr>
<td>32s</td>
<td>No data</td>
</tr>
</tbody>
</table>

An additional parameter, `sleepAddition`, defines the increment value added to the `minSleepTime` parameter at each call until `maxSleepTime` is reached. For example, a setting of 8s would generate a table as follows:

<table>
<thead>
<tr>
<th>Sleep</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>16s</td>
<td>No data</td>
</tr>
<tr>
<td>24s</td>
<td>No data</td>
</tr>
<tr>
<td></td>
<td>Data found</td>
</tr>
<tr>
<td>16s</td>
<td>No data</td>
</tr>
<tr>
<td>24s</td>
<td>No data</td>
</tr>
<tr>
<td>32s</td>
<td>No data</td>
</tr>
<tr>
<td>32s</td>
<td>No data</td>
</tr>
</tbody>
</table>

4.1.9. Troubleshooting Oracle CDC Deployments

The following guides provide information for troubleshooting and addressing problems with Oracle deployments.

- Extractor Slow-down on Single Service

  If when replicating from Oracle, a significant increase in the latency for the extractor within a single service, it may be due to the size of changes and the data not being automatically purged correctly by Oracle.

  The CDC capture tables grow over time, and are automatically purged by Oracle by performing a split on the table partition and releasing the change data from the previous day. In some situations, the purge process is unable to acquire the lock required to partition the table. By default, the purge job does not wait to acquire the lock. To change this behavior, the `DDL_LOCK_TIMEOUT` parameter can be set so that the partition operation waits for the lock to be available. For more information on setting this value, see Oracle `DDL_LOCK_TIMEOUT`.

4.1.9.1. ORA-00257: ARCHIVER ERROR. CONNECT INTERNAL ONLY, UNTIL FREED

Last Updated: 2016-04-20

Condition or Error

It is possible for the Oracle server to get into a state where Tungsten Clustering is online, and with no other errors showing in the log. However, when logging into the Oracle server an error is returned:

ORA-00257: ARCHIVER ERROR. CONNECT INTERNAL ONLY, UNTIL FREED
Causes

- This is a lack of resources within the Oracle server, and not an issue with Tungsten Clustering.

Rectifications

- The issue can be addressed by increasing the logical size of the recovery area, by connecting to the Oracle database as the system user and running the following command:

```
shell> sqlplus sys/oracle as sysdba
SQL> ALTER SYSTEM SET db_recovery_file_dest_size = 80G;
```
Chapter 5. 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 or Chapter 4, Deploying Oracle Extractors to configure the extractor.

5.1. Deploying the Cassandra Applier

Apache Cassandra provides high performance and fault tolerant storage for large, distributed, data sets.

Replication to Cassandra is performed using the batch applier, and operates as follows:

- Data is extracted from the source database into THL.
- When extracting the data from the THL, the 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 Clustering during replication, and the operation type (INSERT, DELETE, etc) as part of the CSV data.
- The CSV data is then loaded into Cassandra 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 Cassandra operation.

Figure 5.1. Topologies: Replicating to Cassandra

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

5.1.1. Preparing for Cassandra 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 Cassandra side, the database and schema must be created using the existing schema definition so that the databases and tables exist within Cassandra.

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 for MySQL extraction, or Chapter 4, Deploying Oracle Extractors for Oracle CDC extraction.
Deploying Appliers

On the Cassandra host, minimal configuration is required. Since there is often no user or authentication support, the main requirements are that Cassandra has been installed, and the `cqlsh` command is available.

The Cassandra applier also makes use of a Ruby component, but since Ruby is a requirement for all Tungsten Clustering installations, Ruby should already exist.

- Download and install the Datastax Cassandra Ruby driver. Ruby Gems can be used for this:

  ```shell
gem install cassandra-driver
  ```

- You need to create tables within Cassandra according to the databases and tables that need to be replicated; the tables are not automatically created for you. From a Tungsten Clustering deployment directory, the `ddlscan` command can be used to identify the existing tables, and create table definitions for use within Cassandra.

To use `ddlscan`, the template for Cassandra 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-5.4.1-41
 cd tungsten-replicator/support/ddlscan/
ddlscan -user tungsten -url 'jdbc:mysql:thin://host1:13306/test' -pass password \
 -template ddl-mysql-cassandra.vn -db test
```

```sql
generated on Wed Jul 12 22:37:41 BST 2017 by ./ddlscan utility of Tungsten
url = jdbc:mysql:thin://ubuntuheterosrc:3306/test
user = root
dbName = test
*/
CREATE KEYSPACE IF NOT EXISTS "test" WITH REPLICATION = {
   'class' : 'SimpleStrategy', 'replication_factor' : 3 } ;
DROP TABLE "test"."all_mysql_types";
CREATE TABLE "test"."all_mysql_types"(
   "my_id" INT ,
   "my_bit" BOOLEAN /* BIT(1) */,
   "my_tinyint" INT ,
   "my_boolean" INT ,
   "my_smallint" INT ,
   "my_mediumint" INT ,
   "my_int" INT ,
   "my_bigint" BIGINT ,
   "my_decimal_10_5" DECIMAL ,
   "my_float" FLOAT ,
   "my_double" DOUBLE ,
   "my_date" DATE ,
   "my_datetime" TIMESTAMP ,
   "my_timestamp" TIMESTAMP ,
   "my_time" TIME ,
   "my_year" DECIMAL ,
   "my_char_10" TEXT ,
   "my_varchar_10" TEXT ,
   "my_tinytext" TEXT ,
   "my_text" TEXT ,
   "my_mediumtext" TEXT ,
   "my_longtext" TEXT ,
   "my_set_def" TEXT ,
   primary key ("my_id")
);
```

The output should be redirected to a file and then used to create tables within Cassandra:

```
ddlscan -user tungsten -url 'jdbc:mysql:thin://host1:13306/test' -pass password \
 -template ddl-mysql-cassandra.vn -db test >test.ddl
```

The output of the command should be checked to ensure that the table definitions are correct.

The file can then be applied to Cassandra:

```
cat test.ddl | cqlsh
```

This generates the table definitions for live data. The process should be repeated to create the table definitions for the staging data by using the staging template:

```
shell
ddlscan -user tungsten -url 'jdbc:mysql:thin://host1:13306/test' -pass password \
 -template ddl-mysql-cassandra-staging.vn -db test >test.ssl-staging
```
Then applied to Cassandra:

```bash
shell> cat test.ddl-staging | cqlsh
```

The process should be repeated for each database that will be replicated.

Once the preparation of the MySQL and Cassandra databases are ready, you can proceed to installing Tungsten Clustering

### 5.1.2. Install Cassandra Applier

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 Install:**

  ```bash
  shell> cd tungsten-replicator-5.4.1-41
  shell> ./tools/tpm configure alpha \
  --enable-batch-service=true
  shell> ./tools/tpm update
  ```

- **For INI Installs:** Add the following to the `/etc/tungsten/tungsten.ini`

  ```ini
  [alpha]
  ...Existing Replicator Config...
  enable-batch-service=true
  shell> tpm update
  ```

1. Unpack the Tungsten Replicator distribution in staging directory:

   ```bash
   shell> tar zxf tungsten-replicator-5.4.1-41.tar.gz
   ```

2. Change into the staging directory:

   ```bash
   shell> cd tungsten-replicator-5.4.1-41
   ```

3. Configure the installation using `tpm`:

   **Show Staging**

   ```bash
   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 \
   ```

   ```bash
   shell> ./tools/tpm configure alpha \
   --topology=master-slave \
   --master=sourcehost \
   --members=localhost \
   --datasource-type=file \
   --replication-user=tungsten \
   --replication-password=secret \
   --batch-enabled=true \
   --batch-load-language=js \
   --batch-load-template=cassandra \
   --property=replicator.datasource.applier.csvType=cassandra \
   --property=replicator.datasource.global.csvType=cassandra \
   --property=replicator.stage.q-to-dbms.blockCommitInterval=5s \
   --property=replicator.stage.q-to-dbms.blockCommitRowCount=1000 \
   --svc-applier-filters=dropstatementdata
   ```

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

   ```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
   ```
Deploying Appliers

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** [397]

  reset [397]

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

- **--user=tungsten** [403]

  user=tungsten [403]

  System User

- **--install-directory=/opt/continuent** [382]

  install-directory=/opt/continuent [382]

  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** [394]

  profile-script=~/.bash_profile [394]

  Append commands to include env.sh in this profile script

- **--skip-validation-check=HostsFileCheck** [335]

  skip-validation-check=HostsFileCheck [335]

  The --skip-validation-check [335] 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** [346], and could be ignored using **--skip-validation-check=MySQLDefaultTableTypeCheck** [335].

  Setting both **--skip-validation-check** [335] and **--enable-validation-check** [333] is equivalent to explicitly disabling the specified check.

```
Deploying Appliers

**skip-validation-check=InstallerMasterSlaveCheck**

The `--skip-validation-check` 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:

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

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

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

- `--skip-validation-check=DatasourceDBPort`

**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:

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

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

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

- `--skip-validation-check=DirectDatasourceDBPort`

**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:

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

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

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

- `--skip-validation-check=ReplicationServicePipelines`

88
Deploying Appliers

skip-validation-check=ReplicationServicePipelines

The `--skip-validation-check` 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`, and could be ignored using `--skip-validation-check=MySQLDefaultTableTypeCheck`.

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

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` [403]

  Replication topology for the dataservice. Valid values are `star,cluster-slave,master-slave,fan-in,clustered,cluster-alias,all-masters,direct`

- `--master=sourcehost` [386]

  The hostname of the master (extractor) within the current service. If the current host does not match this specification, then the deployment will by default be configured as a master/extractor.

- `--members=localhost` [386]

  Hostnames for the dataservice members

- `--datasource-type=file` [375]

  Database type

- `--replication-user=tungsten` [397]

  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` [396]

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

- `--batch-enabled=true` [367]

  Should the replicator service use a batch applier
Deploying Appliers

• \(--\text{batch-load-language}=\text{js} \)[367]

batch-load-language=js [367]

Which script language to use for batch loading

• \(--\text{batch-load-template}=\text{cassandra} \)[367]

batch-load-template=cassandra [367]

Value for the loadBatchTemplate property

• \(--\text{svc-applier-filters}=\text{dropstatementdata} \)[399]

svc-applier-filters=dropstatementdata [399]

Replication service applier filters

4. Install the services:

```
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 5.1.3, “Management and Monitoring of Cassandra Deployments” for more information.

5.1.3. Management and Monitoring of Cassandra Deployments

Monitoring a Cassandra replication scenario requires checking the status of both the master - extracting data from MySQL - and the slave which retrieves the remote THL information and applies it to Cassandra.

```
shell> trepctl status
Processing status command...
NAME   VALUE
 ----   -----  
appliedLastEventId : mysql-bin.000045:0000000000000505;-1
appliedLastSeqno   : 4861
autoRecoveryEnabled: false
autoRecoveryTotal  : 0
channels           : 1
clusterName        : alpha
currentEventId     : mysql-bin.000045:0000000000000505
currentTimeMillis  : 1499956897965
dataServerHost     : ubuntuheterosrc
extensions         : 
host               : ubuntuheterosrc
latestEpochNumber  : 4806
masterConnectUri   : thl://localhost:/
masterListenUri    : thl://ubuntuheterosrc:2112/
maximumStoredSeqNo : 4861
minimumStoredSeqNo : 0
offlineRequests    : NONE
pendingError       : NONE
pendingErrorCode   : NONE
pendingErrorEventId: NONE
pendingErrorSeqno  : -1
pendingExceptionMessage: NONE
relativeLatency    : 111.965
resourcePrecedence : 99
rmPort             : 10088
role               : master
seqnoType          : java.lang.Long
serviceName        : alpha
callServiceName    : alpha
siteName           : default
sourceId           : ubuntuheterosrc
state              : ONLINE
timeInStateSeconds : 1868186.053
timezone           : GMT
transitioningTo    : 
uptimeSeconds      : 1868187.749
useSslConnection   : false
version            : Tungsten Replicator 5.4.1 build 41
```
Deploying Appliers

On the slave, the output of `trepctl` shows the current sequence number and applier status:

```
shell> trepctl status
Processing status command...
NAME                     VALUE
----                     -----  
appliedLastEventId     : mysql-bin.000045:0000000000000505;-1
appliedLastSeqno       : 4861
appliedLatency         : 0.0
autoRecoveryEnabled    : false
autoRecoveryTotal      : 0
channels               : 1
clusterName            : alpha
currentTimeMillis      : 1499896056588
dataServerHost         : cassandra
extensions             : 
host                   : cassandra
latestEpochNumber      : 4806
masterConnectUri       : thl://ubuntuheterosrc:2112/
masterListenUri        : null
maximumStoredSeqNo     : 4861
minimumStoredSeqNo     : 0
offlineRequests        : NONE
pendingError           : NONE
pendingErrorCode       : NONE
pendingErrorEventId    : NONE
pendingErrorSeqno      : -1
pendingExceptionHandler: NONE
pipelineSource         : thl://ubuntuheterosrc:2112/
resourcePrecedence     : 99
rmInurt                : 25550
role                   : slave
seqnoType              : java.lang.Long
serviceName            : alpha
serviceType            : local
simpleServiceName      : alpha
state                  : ONLINE
timeInStateSeconds     : 1272.784
timezone               : GMT
transitioningTo        : 
uptimeSeconds          : 126394.304
useSSLConnection       : false
version                : Tungsten Replicator 5.4.1 build 41
```

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.

5.4. Troubleshooting Cassandra Installations

- Remember that changes to the DDL within the source database are not automatically replicated to Cassandra. Changes to the table definitions, additional tables, or additional databases, must all be updated manually within Cassandra.

- If you get errors similar to:

```
stage_xxx_access_log does not exist
```

When loading into Cassandra, it means that the staging tables have not created correctly. Check the steps for creating the staging tables using `ddlscan` in Section 5.1.1, “Preparing for Cassandra Replication”.

5.2. Deploying the Elasticsearch Applier

Elasticsearch is a highly scalable open-source full-text search and analytics engine. It allows you to store, search, and analyze big volumes of data quickly and in near real time. It is generally used as the underlying engine/technology that powers applications that have complex search features and requirements.

The deployment of Tungsten Clustering to Elasticsearch service is slightly different. There are two parts to the process:

- Service Alpha on the master extracts the information from the MySQL binary log into THL.

- Service Alpha on the slave reads the information from the remote replicator as THL, and applies that to Elasticsearch.
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 Elasticsearch applier, information is extracted from the source database using the row-format, column names and primary keys are identified, and translated to the format supported by Elasticsearch.

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 10.4.8, “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 10.4.30, “PrimaryKey Filter” filter is used to add 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 filter in place, performing update or delete operations require a full table scan to be performed on the target dataserver to determine the record that must be updated.
3. The THL information is then applied to Elasticsearch using the Elasticsearch applier.

The two replication services can operate on the same machine, (See Section 6.2, “Deploying Multiple Replicators on a Single Host”) or they can be installed on two different machines.

5.2.1. Preparing for Elasticsearch 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 for MySQL extraction, or Chapter 4, Deploying Oracle Extractors for Oracle CDC extraction.

During the replication process, data is exchanged from the MySQL database/table/row structure into corresponding Elasticsearch structures, as follows

<table>
<thead>
<tr>
<th>MySQL</th>
<th>Elasticsearch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Index</td>
</tr>
<tr>
<td>Table</td>
<td>Type</td>
</tr>
<tr>
<td>Row</td>
<td>Document</td>
</tr>
</tbody>
</table>

In general, it is easier to understand that a row within the MySQL table is converted into a single document on the Elasticsearch side, and automatically added to index “schema name” with a type of “table name”.

Figure 5.2. Topologies: Replicating to Elasticsearch
Deploying Appliers

For example, the following row within MySQL:

```sql
mysql> select * from messages where id = 99999 ;
*************************** 1. row ***************************
 id: 99999
 msg: Hello Elasticsearch
1 row in set (0.00 sec)
```

Is replicated into the Elasticsearch document:

```json
{
    "_index" : "test",
    "_type" : "messages",
    "_id" : "99999",
    "_version" : 1,
    "found" : true,
    "_source" : {
        "msg" : "Hello Elasticsearch",
        "committime" : "2017-06-23 19:02:22.0",
        "id" : "99999",
        "source_table" : "messages",
        "source_schema" : "test"
    }
}
```

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 Elasticsearch.

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 Elasticsearch 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.

Elasticsearch Host

- Enable networking; by default Elasticsearch is configured to listen only on the localhost (127.0.0.1) IP address. The configuration should be changed to the IP address of your host.

For example, edit the `config/elasticsearch.yml` file and add the following two lines:

```yaml
network.host: _site_
discovery.type: single-node
```

- You may then start Elasticsearch as a daemon:

```bash
bin/elasticsearch -d -p elasticsearch.pid
```

- Check that Elasticsearch is up and running:

```bash
curl http://$HOSTNAME:9200/
```

```json
{
    "name" : "i7DyHKt",
    "cluster_name" : "elasticsearch",
    "cluster_uuid" : "NKL8ZX0RSjyC1HQNAPxJ9w",
    "version" : {
        "number" : "5.4.1",
        "build_hash" : "2cfe0df",
        "build_date" : "2017-05-29T16:05:51.443Z",
        "build_snapshot" : false,
        "lucene_version" : "6.5.1"
    },
    "tagline" : "You Know, for Search"
}
```
5.2.2. Install Elasticsearch Applier

Installation of the Elasticsearch replication requires special configuration of the master and slave hosts so that each is configured for the correct datasource type.

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 Install:

     ```
     shell> cd tungsten-replicator-5.4.1-41
     shell> ./tools/tpm configure alpha
     --repl-svc-extractor-filters=colnames,pkey
     --property=replicator.filter.pkey.addColumnsToDeletes=true
     --property=replicator.filter.pkey.addPkeyToInserts=true
     shell> ./tools/tpm update
     ```

   - For INI Installs: Add the following to `/etc/tungsten/tungsten.ini`

     ```ini
     [alpha]
     ...Existing Replicator Config...
     repl-svc-extractor-filters=colnames,pkey
     property=replicator.filter.pkey.addColumnsToDeletes=true
     property=replicator.filter.pkey.addPkeyToInserts=true
     shell> tpm update
     ```

2. Unpack the Tungsten Replicator distribution in staging directory:

   ```
   shell> tar zxf tungsten-replicator-5.4.1-41.tar.gz
   ```

3. Change into the staging directory:

   ```
   shell> cd tungsten-replicator-5.4.1-41
   ```

4. Configure the installation using `tpm`:

   Show Staging

   ```
   shell> ./tools/tpm configure defaults
   --reset
   --user=tungsten
   --install-directory=/opt/continuent
   --profile-script=~/.bash_profile
   shell> ./tools/tpm configure alpha
   --topology=master-slave
   --master=sourcehost
   --members=localhost
   --datasource-type=elasticsearch
   --replication-user=root
   --replication-password=null
   --replication-port=9200
   --property=replicator.stage.q-to-dbms.filters=casetransform
   --property=replicator.filter.casetransform.to_upper_case=false
   ```

   Show INI

   ```ini
   [defaults]
   user=tungsten
   install-directory=/opt/continuent
   profile-script=~/.bash_profile
   )[alpha]
   topology=master-slave
   master=sourcehost
   members=localhost
   datasource-type=elasticsearch
   replication-user=root
   replication-password=null
   replication-port=9200
   property=replicator.stage.q-to-dbms.filters=casetransform
   property=replicator.filter.casetransform.to_upper_case=false
   ```

   Configuration group `defaults`
Deploying Appliers

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

- **--user=tungsten**
  - user=tungsten
  
  System User

- **--install-directory=/opt/continuent**
  - install-directory=/opt/continuent
  
  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**
  - profile-script=~/.bash_profile
  
  Append commands to include env.sh in this profile script

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**
  - topology=master-slave
  
  Replication topology for the dataservice. Valid values are star, cluster-slave, master-slave, fan-in, clustered, cluster-alias, all-masters, direct.

- **--master=sourcehost**
  - master=sourcehost
  
  The hostname of the master (extractor) within the current service. If the current host does not match this specification, then the deployment will by default be configured as a master/extractor.

- **--members=localhost**
  - members=localhost
  
  Hostnames for the dataservice members

- **--datasource-type=elasticsearch**
  - datasource-type=elasticsearch
  
  Database type

- **--replication-user=root**
  - replication-user=root
  
  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**
  - replication-password=null
  
  The password to be used when connecting to the database using the corresponding **--replication-user**.
Deploying Appliers

- \texttt{--replication-port=9200} [396]

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 Elasticsearch applier slave host(s) only:

\begin{verbatim}
shell> mkdir -p /opt/continuent/share/
shell> cp tungsten-replicator/support/filters-config/convertstringfrommysql.json /opt/continuent/share/
\end{verbatim}

Then, include the following parameters in the configuration:

\begin{verbatim}
property=replicator.stage.remote-to-thl.filters=convertstringfrommysql
property=replicator.filter.convertstringfrommysql.definitionsFile=/opt/continuent/share/convertstringfrommysql.json
\end{verbatim}

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

\begin{verbatim}
shell> ./tools/tpm install
\end{verbatim}

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 5.2.4, "Management and Monitoring of Elasticsearch Deployments" for more information.

5.2.3. Optional Configuration Parameters for Elasticsearch

A number of optional, configurable, properties are available that control how Tungsten Replicator applies and populates information when the data is written into Elasticsearch. The following properties can by set during configuration using \texttt{--property=PROPERTYNAME=value} [334]:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>replicator.applier.dbms.docIdFormat</td>
<td>Determines the format of the document ID used to store the information with Elasticsearch</td>
</tr>
<tr>
<td>replicator.applier.dbms.embedCommitTime</td>
<td>Sets whether the commit time for the source row is embedded into the document</td>
</tr>
<tr>
<td>replicator.applier.dbms.embedSchemaTable</td>
<td>Embed the source schema name and table name in the stored document</td>
</tr>
<tr>
<td>replicator.applier.dbms.ignoreDeleteErrors</td>
<td>Whether a failure to delete an entry should raise an error</td>
</tr>
<tr>
<td>replicator.applier.dbms.ignoreUpdateErrors</td>
<td>Whether a failure to update an existing entry should raise an error</td>
</tr>
<tr>
<td>replicator.applier.dbms.indexName</td>
<td>The string to be used as the index name when the incoming schema name is not used</td>
</tr>
<tr>
<td>replicator.applier.dbms.selfGeneratedId</td>
<td>Determines whether to use a self generated document id or one derived from the primary key information</td>
</tr>
<tr>
<td>replicator.applier.dbms.typeName</td>
<td>The string to use as the index type name when the incoming table is not used</td>
</tr>
<tr>
<td>replicator.applier.dbms.useSchemaAsIndex</td>
<td>Specifies whether the incoming schema name for the data should be used as the Index Name</td>
</tr>
<tr>
<td>replicator.applier.dbms.useTableAsType</td>
<td>Specifies whether the incoming table name for the data should be used as the Index Type</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>replicator.applier.dbms.docIdFormat</td>
<td>Determines the format of the document ID used to store the information with Elasticsearch</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
<tr>
<td>Default</td>
<td>pkey</td>
</tr>
</tbody>
</table>
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 document ID used to store the information into Elasticsearch. For example, when configured to use `tspkeyus`, then the format of the document ID will consist of the schemaname, table name and primary key column information separated by underscores, `SCHEMANAME_TABLENAME_234`.

- `replicator.applier.dbms.embedCommitTime` [97]

| Option  | `replicator.applier.dbms.embedCommitTime` [97]  
|---|---|
| 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:

```
{
"_index" : "test",
"_type" : "messages",
"_id" : "99999",
"_version" : 1,
"found" : true,
"_source" : {
  "msg" : "Hello Elasticsearch",
  "committime" : "2017-06-06 19:09:20.0",
  "id" : "99999",
  "source_table" : "messages",
  "source_schema" : "test"
}
```

- `replicator.applier.dbms.embedSchemaTable` [97]

| Option  | `replicator.applier.dbms.embedSchemaTable` [97]  
|---|---|
| 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`).

```
{
"_index" : "test",
"_type" : "messages",
"_id" : "99999",
"_version" : 1,
"found" : true,
"_source" : {
  "msg" : "Hello Elasticsearch",
```

97
Deploying Appliers

Option: `replicator.applier.dbms.ignoreDeleteErrors` [98]

Description: Whether a failure to delete an entry should raise an error

Value Type: boolean

Default: false

Valid Values:
- false: Raise a failure when deleting records from Elasticsearch fails
- true: Ignore errors when deleting records from Elasticsearch

The `replicator.applier.dbms.ignoreDeleteErrors` determines whether the replicator will fail when trying to delete an entry in the Elasticsearch index entry. This option can be useful when deleting data in an Elasticsearch target and data does not need to guarantee to be deleted.

Option: `replicator.applier.dbms.ignoreUpdateErrors` [98]

Description: Whether a failure to update an existing entry should raise an error

Value Type: boolean

Default: false

Valid Values:
- false: Raise a failure when updating records from Elasticsearch fails
- true: Ignore errors when updating records from Elasticsearch

The `replicator.applier.dbms.ignoreUpdateErrors` determines whether the replicator will fail when trying to update an entry in the Elasticsearch index entry. This option can be useful when updating data in an Elasticsearch target and data does not need to guarantee to be deleted.

Option: `replicator.applier.dbms.indexName` [98]

Description: The string to be used as the index name when the incoming schema name is not used

Value Type: string

Option: `replicator.applier.dbms.selfGeneratedId` [98]

Description: Determines whether to use a self generated document id or one derived from the primary key information

Value Type: boolean

Default: false

Valid Values:
- false: Use the primary key column information as the document ID
- true: Get Elasticsearch to self generate an ID

Option: `replicator.applier.dbms.typeName` [98]

Description: The string to use as the index type name when the incoming table is not used
Deploying Applicators

| Value Type | string |

Specifies the type name to be used when replicator.applier.dbms.useTableAsType has been set to false. The value will be used as the index type HTTP://ELASTIC/INDEXNAME/INDEXTYPE.

replicator.applier.dbms.useTableAsType [99]

| Option | replicator.applier.dbms.useTableAsType [99] |
| Description | Specifies whether the incoming table name for the data should be used as the Index Type |
| Value Type | boolean |
| Default | true |
| Valid Values | false | Do not use the incoming table name as the Elasticsearch index type |
| | true | Use the incoming table name as the Elasticsearch index type |

Changes whether the string used for the index type within the URL [HTTP://ELASTIC/INDEXNAME/INDEXTYPE]. If set to false, the replicator.applier.dbms.typeName must be used to specify the index type.

replicator.applier.dbms.useSchemaAsIndex [99]

| Option | replicator.applier.dbms.useSchemaAsIndex [99] |
| Description | Specifies whether the incoming schema name for the data should be used as the Index Name |
| Value Type | boolean |
| Default | true |
| Valid Values | false | Do not use the incoming schema name as the Elasticsearch index name |
| | true | Use the incoming schema name as the Elasticsearch index name |

Changes whether the string used for the index name within the URL [HTTP://ELASTIC/INDEXNAME/INDEXTYPE]. If set to false, the replicator.applier.dbms.typeName must be used to specify the index type.

5.2.4. Management and Monitoring of Elasticsearch 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
clusternname        : 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
```
Deploying Appliers

The replicator service operates just the same as a standard master service of a typical MySQL replication service.

The Elasticsearch applier service can be accessed either remotely from the master:

```
shell> trepctl -host elastichost status
...
```

Or locally on the Elasticsearch 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      : elastichost
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
rmPort             : 10000
role               : slave
seqnoType          : java.lang.Long
serviceName         : alpha
serviceType         : local
simpleServiceName   : alpha
siteName            : default
sourceId            : elastichost
state               : ONLINE
timeInStateSeconds  : 177783.343
transitioningTo     : 
uptimeSeconds       : 180631.276
useSSLConnection    : false
version             : Tungsten Replicator 5.4.1 build 41
Finished status command...
```

Monitoring the status of replication between the master and slave is also the same. The `appliedLastSeqno` still indicates the sequence number that has been applied to Elasticsearch, and the event ID from Elasticsearch can still be identified from `appliedLastEventId`.

Sequence numbers between the two hosts should match, as in a master/slave deployment, but due to the method used to replicate, the applied latency may be higher.

To check for information within Elasticsearch, use the `curl` command-line client:

```
shell> curl http://$HOSTNAME:9200/_cat/indices?v
Health status index uid prt rep docs.count docs.deleted store.size pri.store.size
yellow open tungsten_alpha dpdRRQ_L7_5AMA8tLUX6g 5 2 0 13kb 13kb
```

To test replication, execute typical commands in MySQL and use the `curl` command-line client to read them back:
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 master database.

Deployment of the Hadoop replication is similar to other heterogeneous installations; two separate installations are created:

- Service Alpha on the master extracts the information from the MySQL binary log into THL.
- Service Alpha on the slave reads the information from the remote replicator as THL, applying it to Hadoop. The applier works in two stages:
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.

### 5.3.1. Hadoop Replication Operation

The Hadoop applier makes use of the JavaScript based batch loading system (see Section 6.5.4, "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 5.4, "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 slave 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 master, plus metadata describing the operation for each row, the sequence number, and then the full row information.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Sequence No</th>
<th>Unique Row</th>
<th>Commit TimeStamp</th>
<th>Table-specific primary key</th>
<th>Table-column</th>
</tr>
</thead>
<tbody>
<tr>
<td>I (Insert) or D (Delete)</td>
<td><code>seqno [503]</code> that generated this row</td>
<td>Unique row ID within the batch</td>
<td>The commit time-stamp of the original transaction, which can be used for partitioning</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For example, the MySQL row:

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

Is represented within the staging files generated as:
Deploying Appliances

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 \x01 character for fields. For more information on these fields and how they can be configured, see Section 6.5.7, “Supported CSV Formats”.

On the Hadoop host, information is stored into a number of locations within the HDFS during the data transfer:

Table 5.2. Hadoop Replication Directory Locations

<table>
<thead>
<tr>
<th>Directory/File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/user/USERNAME</td>
<td>Top-level directory for Tungsten Replicator information, using the configured replication user.</td>
</tr>
<tr>
<td>/user/tungsten/metadata</td>
<td>Location for metadata related to the replication operation</td>
</tr>
<tr>
<td>/user/tungsten/metadata/alpha</td>
<td>The directory [named after the servicename of the replicator service] that holds service-specific metadata</td>
</tr>
<tr>
<td>/user/tungsten/staging</td>
<td>Directory of the data transferred</td>
</tr>
<tr>
<td>/user/tungsten/staging/service-name</td>
<td>Directory of the data transferred from a specific servicename.</td>
</tr>
<tr>
<td>/user/tungsten/staging/service-name/database</td>
<td>Directory of the data transferred specific to a database.</td>
</tr>
<tr>
<td>/user/tungsten/staging/service-name/database/tablename</td>
<td>Directory of the data transferred specific to a table.</td>
</tr>
<tr>
<td>/user/tungsten/staging/service-name/database/tablename/tablename-###.csv</td>
<td>Filename of a single file of the data transferred for a specific table and database.</td>
</tr>
</tbody>
</table>

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 1279236 2014-01-13 06:58 /user/tungsten/staging/alpha/hadoop/chicago/chicago-10.csv
-rw-r--r-- 3 cloudera cloudera 10274189 2014-01-13 08:33 /user/tungsten/staging/alpha/hadoop/chicago/chicago-103.csv
-rw-r--r-- 3 cloudera cloudera 1275832 2014-01-13 08:33 /user/tungsten/staging/alpha/hadoop/chicago/chicago-104.csv
-rw-r--r-- 3 cloudera cloudera 1275411 2014-01-13 08:33 /user/tungsten/staging/alpha/hadoop/chicago/chicago-105.csv
-rw-r--r-- 3 cloudera cloudera 10370471 2014-01-13 08:33 /user/tungsten/staging/alpha/hadoop/chicago/chicago-113.csv
-rw-r--r-- 3 cloudera cloudera 1279435 2014-01-13 08:33 /user/tungsten/staging/alpha/hadoop/chicago/chicago-114.csv
-rw-r--r-- 3 cloudera cloudera 2544862 2014-01-13 06:58 /user/tungsten/staging/alpha/hadoop/chicago/chicago-12.csv
-rw-r--r-- 3 cloudera cloudera 11694282 2014-01-13 08:33 /user/tungsten/staging/alpha/hadoop/chicago/chicago-123.csv
-rw-r--r-- 3 cloudera cloudera 1279872 2014-01-13 08:34 /user/tungsten/staging/alpha/hadoop/chicago/chicago-124.csv
-rw-r--r-- 3 cloudera cloudera 2572481 2014-01-13 08:34 /user/tungsten/staging/alpha/hadoop/chicago/chicago-126.csv
-rw-r--r-- 3 cloudera cloudera 9073627 2014-01-13 08:34 /user/tungsten/staging/alpha/hadoop/chicago/chicago-133.csv
-rw-r--r-- 3 cloudera cloudera 1279788 2014-01-13 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.

### 5.3.2. 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:

<table>
<thead>
<tr>
<th>MySQL</th>
<th>Hadoop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Directory</td>
</tr>
<tr>
<td>Table</td>
<td>Hive-compatible Character-Separated Text file</td>
</tr>
<tr>
<td>Row</td>
<td>Line in the text file, fields terminated by character \x01</td>
</tr>
</tbody>
</table>

#### 5.3.2.1. 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 master, 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 slave to a relay host is not supported. One host within the Hadoop cluster must be chosen to act as the slave.

The prerequisites for a standard Tungsten Clustering 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
hadoop fs -chmod 700 /user/tungsten
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.

5.3.2.2. Schema Generation

In order to access the generated tables, both staging and the final tables, it is necessary to create a schema definition. The `ddiscan` 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
discan -user tungsten -url 'jdbc:mysql:thin://host1:13306/test' -pass password -template ddl-mysql-hive-0.10.vm -db test
```

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
discan -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
```
Deploying Appliers

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 `\010` field separator and `\012` (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 5.3.4.3, "Accessing Generated Tables in Hive".

5.3.3. Replicating into Kerberos Secured HDFS

Added in 5.3.0. Support for Kerberos based apply was added in Tungsten Replicator 5.3.0.

For replicating into HDFS where Kerberos support has been enabled, the `hadoop_kerberos.js` watch 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` to enable the new batch load script.

5.3.4. Install Hadoop Replication

Installation of the Hadoop replication consists of multiple stages:

1. Install the Master replicator following the appropriate extractor guides
   - Chapter 3, Deploying MySQL Extractors
   - Chapter 4, Deploying Oracle Extractors
2. Install the Slave replicator which will apply information to the target Hadoop environment.
3. Once the installation of the Master and Slave components have been completed, materialization of tables and views.

5.3.4.1. Slave Replicator Service

The slave replicator service reads information from the THL of the master and applies this to a local instance of Hadoop.

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

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-5.4.1-41
     shell> /tools/tpm configure alpha \n     --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-5.4.1-41.tar.gz
```

3. Change into the staging directory:

```
shell> cd tungsten-replicator-5.4.1-41
```

4. Configure the installation using `tpm`:

Show Staging

Show INI

```
shell> ./tools/tpm configure defaults \
   --reset \n   --user=tungsten \n   --install-directory=/opt/continuent \n   --profile-script=~/bash_profile \n   --skip-validation-check=HostsFileCheck \n   --skip-validation-check=InstallerMasterSlaveCheck \n   --skip-validation-check=DatasourceDBPort \n   --skip-validation-check=DirectDatasourceDBPort \n   --skip-validation-check=ReplicationServicePipelines

shell> ./tools/tpm configure alpha \
   --master=host1 \n   --members=host2 \n   --property=replicator.datasource.global.csvType=hive \n   --property=replicator.stage.q-to-dbms.blockCommitInterval=1s \n   --property=replicator.stage.q-to-dbms.blockCommitRowCount=1000 \n   --replication-password=secret \n   --replication-user=tungsten \n   --batch-enabled=true \n   --batch-load-language=js \n   --batch-load-template=hadoop \n   --datasource-type=file

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

```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

[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` [397]

  reset [397]

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

- `--user=tungsten` [403]

  user=tungsten [403]
Deploying Appliers

System User

• **--install-directory=/opt/continuent**

install-directory=/opt/continuent

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**

profile-script=~/.bash_profile

Append commands to include env.sh in this profile script

• **--skip-validation-check=HostsFileCheck**

skip-validation-check=HostsFileCheck

The **--skip-validation-check** 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:

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

The check in this case is MySQLDefaultTableTypeCheck, and could be ignored using **--skip-validation-check=mysqldefaulttablertypecheck**.
```

Setting both **--skip-validation-check** and **--enable-validation-check** is equivalent to explicitly disabling the specified check.

• **--skip-validation-check=InstallerMasterSlaveCheck**

skip-validation-check=InstallerMasterSlaveCheck

The **--skip-validation-check** 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:

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

The check in this case is MySQLDefaultTableTypeCheck, and could be ignored using **--skip-validation-check=mysqldefaulttablertypecheck**.
```

Setting both **--skip-validation-check** and **--enable-validation-check** is equivalent to explicitly disabling the specified check.

• **--skip-validation-check=DatasourceDBPort**

skip-validation-check=DatasourceDBPort

The **--skip-validation-check** 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` [346], and could be ignored using `--skip-validation-check=MySQLDefaultTableTypeCheck` [335].

Setting both `--skip-validation-check` [335] and `--enable-validation-check` [333] is equivalent to explicitly disabling the specified check.

- `--skip-validation-check=DirectDatasourceDBPort` [335]
- `skip-validation-check=DirectDatasourceDBPort` [335]

The `--skip-validation-check` [335] 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` [346], and could be ignored using `--skip-validation-check=MySQLDefaultTableTypeCheck` [335].

Setting both `--skip-validation-check` [335] and `--enable-validation-check` [333] is equivalent to explicitly disabling the specified check.

- `--skip-validation-check=ReplicationServicePipelines` [335]
- `skip-validation-check=ReplicationServicePipelines` [335]

The `--skip-validation-check` [335] 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` [346], and could be ignored using `--skip-validation-check=MySQLDefaultTableTypeCheck` [335].

Setting both `--skip-validation-check` [335] and `--enable-validation-check` [333] is equivalent to explicitly disabling the specified check.

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-host` [386]
Deploying Appliers

master=host1

The hostname of the master (extractor) within the current service. If the current host does not match this specification, then the deployment will by default be configured as a master/extractor.

- --members=host2

members=host2

Hostnames for the dataservice members

- --replication-password=secret

replication-password=secret

The password to be used when connecting to the database using the corresponding --replication-user.

- --replication-user=tungsten

replication-user=tungsten

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

batch-enabled=true

Should the replicator service use a batch applier

- --batch-load-language=js

batch-load-language=js

Which script language to use for batch loading

- --batch-load-template=hadoop

batch-load-template=hadoop

Value for the loadBatchTemplate property

- --datasource-type=file

datasource-type=file

Database type

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 service has been installed it can be monitored using the trepctl command. See Section 5.3.4.4, "Management and Monitoring of Hadoop Deployments" for more information. If there are problems during installation, see Section 5.3.4.5, "Troubleshooting Hadoop Replication".

5.3.4.2. 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

5.3.4.3. Accessing Generated Tables in Hive

If not already completed, the schema generation process described in Section 5.3.2.2, “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 Goes Blind (#1.4)
I 10 6 57480 All in the Family 1971 Archie Goes Too Far (#1.7)
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 (#1.15)
I 10 9 57483 All in the Family 1971 Archie in the Lock-Up (#1.2)
I 10 10 57484 All in the Family 1971 Archie Is Branded (#3.20)
```

5.3.4.4. 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 master (extractor) service:

```
shell> trepctl status
Processing status command...
NAME VALUE
---- ----
appliedLastEventId : mysql-bin.000023:0000000505545003;0
appliedLastSeqno : 10992
appliedLatency : 43.764
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
siteName : default
sourceId : host1
state : ONLINE
timeInStateSeconds : 165845.474
transitioningTo :
uptimeSeconds : 165850.047
useSslConnection : False
version : Tungsten Replicator 5.4.1 build 41
Finished status command...
```

When monitoring, the primary concerns beyond identifying and copying with any errors is to monitor the applied latency. Larger numbers for applied latency generally indicate 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.
Deploying Appliers

- 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.

5.3.4.5. 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.

5.3.4.5.1. Errors Reading/Writing commitseqno.0 File

During initial installation, or when starting up replication, the replicator may report that the commitseqno.0 cannot 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
dir -ls /user/tungsten/metadata
```

- 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:

  ```json
  
  "appliedLatency": "0",
  "epochNumber": "0",
  "fragno": "0",
  "shardId": "dna",
  "seqno": "8",
  "eventId": "mysql-bin.000015:0000000000103156;0",
  "extractedTstamp": "1389706078000",
  "lastFrag": "true",
  "sourceId": "host1"
  
  
```

- Try deleting the commitseqno.0 file and placing the replicator online:

  ```shell
dir -rm /user/tungsten/metadata/commitseqno.0
dir trepctl online
```

5.3.4.5.2. 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 master and CSV files will be recreated and applied into HDFS.

1. On the Slave, put the replicator offline:

   ```shell
trepctl offline
```

2. Remove the THL files from the slave:

   ```shell
trepctl reset -thl
```

3. Remove the staging CSV files replicated into Hadoop:

   ```shell
dir -rm /user/tungsten/staging
```

4. Reset the restart position:

   ```shell
dir /opt/continuent/tungsten/tungsten-replicator/data/commitseqno.0
```

Replace alpha and /opt/continuent with the corresponding service name and installation location.

5. Restart replication on the slave; this will start to recreate the THL files from the MySQL binary log:
5.4. 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 Clustering 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 Clustering to Kafka service is slightly different. There are two parts to the process:

- Service Alpha on the master extracts the information from the MySQL binary log into THL.
- Service Alpha on the slave reads the information from the remote replicator as THL, and applies that 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 10.4.8, "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 Section 10.4.30, "PrimaryKey Filter" filter is used to add primary key information to row-based replication data.
4. 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**, when the replicator goes online it may resend rows and messages.

The two replication services can operate on the same machine, (See Section 6.2, "Deploying Multiple Replicators on a Single Host") or they can be installed on two different machines.

### 5.4.1. 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 for MySQL extraction, or Chapter 4, Deploying Oracle Extractors for Oracle CDC extraction.

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:

```sql
mysql> select * from messages where id = 99999
+---------------------+-----------------------+
| id                  | msg                   |
+---------------------+-----------------------+
| 99999               | Hello Kafka           |
+---------------------+-----------------------+
1 row in set (0.00 sec)
```

Is replicated into Kafka as a Kafka message using the topic **test_msg**:

```json
{"_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 actual 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 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 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.
Deploying Appliers

5.4.2. Install Kafka Applier

Installation of the Kafka replication requires special configuration of the master and slave hosts so that each is configured for the correct datasource type.

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 Install:

     ```
     cd tungsten-replicator-5.4.1-41
     ./tools/tpm configure alpha
     --repl-svc-extractor-filters=colnames,pkey
     --property=replicator.filter.pkey.addColumnsToDeletes=true
     --property=replicator.filter.pkey.addPkeyToInserts=true
     ./tools/tpm update
     ```

   • For INI Installs: Add the following the `/etc/tungsten/tungsten.ini`

     ```ini
     [alpha]
     ...Existing Replicator Config...
     repl-svc-extractor-filters=colnames,pkey
     property=replicator.filter.pkey.addColumnsToDeletes=true
     property=replicator.filter.pkey.addPkeyToInserts=true
     ```

     ```
     tpm update
     ```

2. Unpack the Tungsten Replicator distribution in staging directory:

   ```
   tar zxf tungsten-replicator-5.4.1-41.tar.gz
   ```

3. Change into the staging directory:

   ```
   cd tungsten-replicator-5.4.1-41
   ```

4. Configure the installation using `tpm`:

   Show Staging

   ```
   ./tools/tpm configure defaults
   --reset
   --install-directory=/opt/continuent
   --profile-script=~/.bash_profile
   ```

   Show INI

   ```
   ./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
   ```

   ```
   vi /etc/tungsten/tungsten.ini
   ```

   ```ini
   [defaults]
   install-directory=/opt/continuent
   profile-script=~/.bash_profile
   ```

   ```ini
   [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 [397]
  reset [397]

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

• --install-directory=/opt/continuent [382]
  install-directory=/opt/continuent [382]

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 [394]
  profile-script=~/.bash_profile [394]

Append commands to include env.sh in this profile script

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 [386]
  master=sourcehost [386]

The hostname of the master (extractor) within the current service. If the current host does not match this specification, then the deployment will by default be configured as a master/extractor.

• --members=localhost [386]
  members=localhost [386]

Hostnames for the dataservice members

• --datasource-type=kafka [375]
  datasource-type=kafka [375]

Database type

• --replication-user=root [397]
  replication-user=root [397]

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 [396]
  replication-password=null [396]

The password to be used when connecting to the database using the corresponding --replication-user [397].

• --replication-port=9092 [396]
  replication-port=9092 [396]

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 slave host(s) only:

```bash
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

```bash
property=replicator.stage.remote-to-thl.filters=convertstringfrommysql
```
6. Once the prerequisites and configuring of the installation has been completed, the software can be installed:

```shell
deployer (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 5.4.3, “Management and Monitoring of Kafka Deployments” for more information.

### 5.4.2.1. 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`:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>replicator.applier.dbms.embedCommitTime</code></td>
<td>Sets whether the commit time for the source row is embedded into the document</td>
</tr>
<tr>
<td><code>replicator.applier.dbms.embedSchemaTable</code></td>
<td>Embed the source schema name and table name in the stored document</td>
</tr>
<tr>
<td><code>replicator.applier.dbms.keyFormat</code></td>
<td>Determines the format of the message ID</td>
</tr>
<tr>
<td><code>replicator.applier.dbms.requireacks</code></td>
<td>Defines whether when writing messages to the Kafka cluster, how many acknowledgements from Kafka nodes is required</td>
</tr>
<tr>
<td><code>replicator.applier.dbms.retrycount</code></td>
<td>The number of retries for sending each message</td>
</tr>
<tr>
<td><code>replicator.applier.dbms.zookeeperString</code></td>
<td>Connection string for Zookeeper, including hostname and port</td>
</tr>
</tbody>
</table>

```shell
deployer (tools/tpm) install
```

Embeds the commit time of the source database row into the document information:

```json
{
   "_seqno" : "4865",
   "_source_table" : "msg",
   "_committime" : "2017-07-13 15:30:37.0",
   "_source_schema" : "test",
   "record" : {
     "msg" : "Hello Kafka",
     "id" : "2384726"
   },
   "_optype" : "INSERT"
}
```

```shell
deployer (tools/tpm) install
```

Embeds the source schema name and table name into the stored document information:

```json
{
   "_seqno" : "4865",
   "_source_table" : "msg",
   "_committime" : "2017-07-13 15:30:37.0",
   "_source_schema" : "test",
   "record" : {
     "msg" : "Hello Kafka",
     "id" : "2384726"
   },
   "_optype" : "INSERT"
}
```

```shell
deployer (tools/tpm) install
```
Deploying Appliers

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.keyFormat**
  
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
<th>Default</th>
<th>Valid Values</th>
<th>Value Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>replicator.applier.dbms.keyFormat</td>
<td>Determines the format of the message ID</td>
<td>string</td>
<td>pkey</td>
<td>pkey</td>
<td>Combine the primary key column values into a single string</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>pkeyus</td>
<td>Combine the primary key column values into a single string joined by an underscore character</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>tspkey</td>
<td>Combine the schema name, table name, and primary key column values into a single string joined by an underscore character</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>tspkeyus</td>
<td>Combine the schema name, table name, and primary key column values into a single string</td>
</tr>
</tbody>
</table>

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**
  
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
<th>Default</th>
<th>Valid Values</th>
<th>Value Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>replicator.applier.dbms.requireacks</td>
<td>Defines whether when writing messages to the Kafka cluster, how many acknowledgements from Kafka nodes is required</td>
<td>string</td>
<td>all</td>
<td>1</td>
<td>Only the lead host should acknowledge receipt of the message</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>all</td>
<td>All nodes should acknowledge receipt of the message</td>
</tr>
</tbody>
</table>

Sets the acknowledgement counter for sending messages into the Kafka queue.

- **replicator.applier.dbms.retrycount**
  
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
<th>Default</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>replicator.applier.dbms.retrycount</td>
<td>The number of retries for sending each message</td>
<td>number</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Determines the number of times the message will attempt to be sent before failure.

- **replicator.applier.dbms.zookeeperString**
  
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
<th>Default</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>replicator.applier.dbms.zookeeperString</td>
<td>Connection string for Zookeeper, including hostname and port</td>
<td>string</td>
<td>${replicator.global.db.host}:2181</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5.4.3. 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
shell> trepctl status
appliedLastEventId     : mysql-bin.000008:0000000000412301:0
appliedLastSeqno       : 1296
appliedLatency         : 16.253
autoRecoveryEnabled    : false
autoRecoveryTotal      : 0
channels               : 1
clusterName            : alpha
currentEventId         : mysql-bin.000008:0000000000412301:0
currentTimeMillis      : 1377098139212
dataServerHost         : mysqlhost
extensions             :
host                   : kafka
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         : /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 5.4.1 build 41
```

The replicator service operates just the same as a standard master service of a typical MySQL replication service.

The Kafka applier service can be accessed either remotely from the master:

```shell
shell> trepctl -host kafka status
```

Or locally on the Kafka host:

```shell
shell> trepctl status
```

Finished status command...
Monitoring the status of replication between the master and slave 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 master/slave 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 [391] 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 [391] setting is enabled, or that you are using --enable-heterogeneous-service [380].

5.5. 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 master extracts the information from the MySQL binary log into THL.
- Service Alpha on the slave reads the information from the remote replicator as THL, and applies that to MongoDB.

Figure 5.6. 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 10.4.8, “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 6.2, “Deploying Multiple Replicators on a Single Host”) or they can be installed on two different machines.

5.5.1. 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 5.5.4, “Install MongoDB Atlas Applier”

5.5.2. 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 for MySQL extraction, or Chapter 4, Deploying Oracle Extractors for Oracle CDC extraction.

During the replication process, data is exchanged from the MySQL database/table/row structure into corresponding MongoDB structures, as follows:

<table>
<thead>
<tr>
<th>MySQL</th>
<th>MongoDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Database</td>
</tr>
<tr>
<td>Table</td>
<td>Collection</td>
</tr>
<tr>
<td>Row</td>
<td>Document</td>
</tr>
</tbody>
</table>

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

+----------+-------------------+----------+---------+-------------+----------+---------+--------+---------+-------------+-------------------+----------+-------+-------------------+---------+----------+---------+----------+---------+----------+-------------------+----------+
| recipeid | title             | subtitle | servings| active  | parid       | userid   | rating  | cumrating| createdate | modifydate          | createdate | updatedate| createdate          | updatedate | createdate| updatedate| createdate| updatedate| createdate|
|----------|-------------------+----------+---------+---------+------------+----------+---------+----------+-----------+-------------------+----------+---------+-------------------+-----------+----------+-----------+----------+----------+-----------|
| 1085     | Creamy egg and leek special |          | 4       | 1        | 0          | 0        | 0       | 0         | 0         | (6.00 sec)         |          |          | (6.00 sec)         |           |          |           |          |          |           |
+----------+-------------------+----------+---------+---------+------------+----------+---------+----------+-----------+-------------------+----------+---------+-------------------+-----------+----------+-----------+----------+----------+-----------|
```

Is replicated into the MongoDB document:

```json
{
   "id": ObjectId("5212335b84ae46ce07e427c3"),
   // other fields...
}
```
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 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 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.

5.5.3. 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 5.5.4, “Install MongoDB Atlas Applier”

Installation of the MongoDB replication requires special configuration of the master and slave hosts so that each is configured for the correct datasource type.

To configure the slave 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:

```bash
shell> cd tungsten-replicator-5.4.1-41
shell> ./tools/tpm configure alpha \
    --enable-heterogeneous-master=true
shell> ./tools/tpm update
```

   • For INI installs: Add the following the /etc/tungsten/tungsten.ini

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

```bash
shell> tpm update
```

2. Unpack the Tungsten Replicator distribution in staging directory:

```bash
shell> tar zxf tungsten-replicator-5.4.1-41.tar.gz
```
3. Change into the staging directory:

```
shell> cd tungsten-replicator-5.4.1-41
```

4. Configure the installation using `tpm`:

**Show Staging**

**Show INI**

```
shell> ./tools/tpm configure defaults \
    --reset \n    --install-directory=/opt/continuent \n    --profile-script=~/.bash_profile \n    --disable-security-controls=true
shell> ./tools/tpm configure alpha \
    --master=sourcehost \n    --members=localhost \n    --datasource-type=mongodb \n    --replication-user=tungsten \n    --replication-password=secret \n    --svc-applier-filters=dropstatementdata \n    --role=slave \n    --replication-port=27017
```

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

[.defaults]
install-directory=/opt/continuent
profile-script=~/.bash_profile
disable-security-controls=true

[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` [397]
  
  reset [397]

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

- `--install-directory=/opt/continuent` [382]

  install-directory=/opt/continuent [382]

  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` [394]

  profile-script=~/.bash_profile [394]

  Append commands to include env.sh in this profile script

- `--disable-security-controls=true` [378]

  disable-security-controls=true [378]

  Disables all forms of security, including SSL, TLS and authentication

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`
  
  The hostname of the master (extractor) within the current service. If the current host does not match this specification, then the deployment will by default be configured as a master/extractor.

- `--members=localhost`
  
  Hostnames for the dataservice members

- `--datasource-type=mongodb`
  
  Database type

- `--replication-user=tungsten`
  
  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`
  
  The password to be used when connecting to the database using the corresponding `--replication-user`

- `--svc-applier-filters=dropstatementdata`
  
  Replication service applier filters

- `--role=slave`
  
  What is the replication role for this service?

- `--replication-port=27017`
  
  The network port used to connect to the database server. The default port used depends on the database being configured.

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 5.5.5, “Management and Monitoring of MongoDB Deployments” for more information.

5.5.4. 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 standard MongoDB, please refer to the following section: Section 5.5.3, “Install MongoDB Applier”

Installation of the MongoDB replication requires special configuration of the master and slave hosts so that each is configured for the correct datasource type.

To configure the slave 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:

   ```
   cd tungsten-replicator-5.4.1-41
   ./tools/tpm configure alpha \
   --enable-heterogeneous-master=true
   ./tools/tpm update
   ```

   • For INI installs: Add the following the `/etc/tungsten/tungsten.ini`

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

2. Unpack the Tungsten Replicator distribution in staging directory:

   ```
   tar zxf tungsten-replicator-5.4.1-41.tar.gz
   ```

3. Change into the staging directory:

   ```
   cd tungsten-replicator-5.4.1-41
   ```

4. Configure the installation using `tpm`:

   Show Staging

   ```
   ./tools/tpm configure defaults \
   --reset \
   --install-directory=/opt/continuent \
   --profile-script=~/.bash_profile \
   --disable-security-controls=false \
   --rmi-ssl=true \
   --thl-ssl=true \
   --rmi-authentication=true
   ```

   ```
   ./tools/tpm configure alpha \
   --master=sourcehost \
   --members=localhost \
   --datasource-type=mongodb \
   --replication-user=tungsten \
   --replication-password=secret \
   --svc-applier-filters=dropstatementdata \
   --role=slave \
   --property=replicator.applier.dbms.connectString=mongodb+srv://${replicator.global.db.user}:${replicator.global.db.password}@${replicator.global.db.host}/?retryWrites=true&w=majority
   ```

   ```
   vi /etc/tungsten/tungsten.ini
   ```

   **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.
**DeployingAppliers**

- **--reset**

  reset

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

- **--install-directory=/opt/continuent**

  install-directory=/opt/continuent

  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**

  profile-script=~/.bash_profile

  Append commands to include env.sh in this profile script

- **--disable-security-controls=false**

  disable-security-controls=false

  Disables all forms of security, including SSL, TLS and authentication

- **--rmi-ssl=true**

  rmi-ssl=true

  Enable SSL encryption of RMI communication on this host

- **--thl-ssl=true**

  thl-ssl=true

  Enable SSL encryption of THL communication for this service

- **--rmi-authentication=true**

  rmi-authentication=true

  Enable RMI authentication for the services running on this host

**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**

  master=sourcehost

  Thehostname of the master (extractor) within the current service. If the current host does not match this specification, then the deployment will by default be configured as a master/extractor.

- **--members=localhost**

  members=localhost

  Hostnames for the dataservice members

- **--datasource-type=mongodb**

  datasource-type=mongodb

  Database type

- **--replication-user=tungsten**

  replication-user=tungsten
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`

  The password to be used when connecting to the database using the corresponding `replication-user`

- `svc-applier-filters=dropstatementdata`

  Replication service applier filters

- `role=slave`

  What is the replication role for this service?

- `replication-host=atlasendpoint.mongodb.net`

  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 slave (relay) connection.

- `replication-port=27017`

  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`

  The `property` 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. 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.

MongoDB Atlas runs with SSL enabled by default and therefore we need to configure the replicator to recognise this.

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

   ```plaintext
   -----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 certificate 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
   ```

Once the replicators have started, the status of the service can be checked using `trepctl`. See Section 5.5.5, “Management and Monitoring of MongoDB Deployments” for more information.

### 5.5.5. 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
trepcctl status
```

```
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
events : 
latestEpochNumber : 1286
masterConnectUri : thl://localhost:10000/master
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
storedProcedure : java.lang.Long
serviceName : alpha
serviceType : local
simpleServiceName : alpha
siteName : default
sourceId : host1
state : ONLINE
timeInStateSeconds : 177443.948
transitioningTo : 
timeSinceLastTransaction : 177461.483
version : Tungsten Replicator 5.4.1 build 41
Finished status command...
```
Deploying Appliers

The replicator service operates just the same as a standard master service of a typical MySQL replication service.

The MongoDB applier service can be accessed either remotely from the master:

```sh
trepctl -host host2 status
```

Or locally on the MongoDB host:

```sh
trepctl status
```

```
NAME                     VALUE
----                     -----  
appliedLastEventId      mysql-bin.088008:088000000908412381:0
appliedLastSeqno        1296   
appliedLatency          10.253 
channels                1
clusterName             alpha  
currentEventId          XOFF   
currentTimeMillis       1377098139212
dataServerHost          host2  
extensions              
latestEpochNumber       1286   
masterConnectUri        thl://host1:2112/
masterListenUri         null   
maximumStoredSeqNo      1296   
mongoDBVersion          2.2.4   
mongoDBShellVersion     2.2.4   
minimumStoredSeqNo      0
offlineRequests          XNONE  
pendingError            XNONE  
pendingErrorCode        XNONE  
pendingErrorEventId     XNONE  
pendingErrorSeqno       -1
pendingExceptionMessage XNONE  
pipelineSource          thl://host1:2112/
relativeLatency         377771.212
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 5.4.1 build 41
```

Monitoring the status of replication between the master and slave 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 master/slave 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:

```sh
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:

```sh
> 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
```
Collection counts should match the row count of the source tables:

```bash
> db.recipe.count()
9289
```

The `db.collection.find()` command can be used to list the documents within a given collection.

```bash
> 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",
  "cureating" : "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",
  "cureating" : "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:

```bash
"title" : "{B@7084a5c"
```

It probably indicates that UTF8 and/or `--mysql-use-bytes-for-string=false` options were not used during installation. The configuration can be updated using `tpm` to address this issue.

### 5.6. Deploying the MySQL Applier

Deploying the MySQL applier is the most straightforward 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 5.7. Topologies: Replicating to MySQL

The slave 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 6.2, “Deploying Multiple Replicators on a Single Host”)

5.6.1. 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 for MySQL extraction, or Chapter 4, Deploying Oracle Extractors for Oracle CDC extraction.

• 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
Deploying Appliers

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`):

```text
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`):

```text
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 5.6.2, “Prepare Amazon RDS/Amazon Aurora”

If your target is a hosted MySQL environment, proceed to Section 5.6.3, “Install MySQL Applier”

5.6.2. 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 slave 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 slave 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”.

5.6.3. Install MySQL Applier

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

To configure the slave 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-5.4.1-41.tar.gz
```

- Change into the staging directory:

```shell
cd tungsten-replicator-5.4.1-41
```

- Use the appropriate template config for your target

  - Section 5.6.3.1, “Local and Remote MySQL Targets”
  - Section 5.6.3.2, “Amazon RDS and Amazon Aurora Targets”

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

```shell
toolkit/tungsten-configure
```

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.
Deploying Appliers

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

### 5.6.3.1. Local and Remote MySQL Targets

- Configure the installation using `tpm`:

  ```bash
  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
  
  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
  
  [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` [397]
    
    reset [397]
    
    For staging configurations, deletes all pre-existing configuration information between updating with the new configuration values.
  - `--install-directory=/opt/continuent` [382]
    
    install-directory=/opt/continuent [382]
    
    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` [403]
    
    user=tungsten [403]
    
    System User
  - `--mysql-allow-intensive-checks=true` [389]
    
    mysql-allow-intensive-checks=true [389]
    
    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` [394]
    
    profile-script=~/.bash_profile [394]
Deploying Appliers

Append commands to include env.sh in this profile script

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 [386]**
  
  `master=sourcehost [386]`

  The hostname of the master (extractor) within the current service. If the current host does not match this specification, then the deployment will by default be configured as a master/extractor.

- • **--members=localhost,sourcehost [386]**
  
  `members=localhost,sourcehost [386]`

  Hostnames for the dataservice members

- • **--datasource-type=mysql [375]**
  
  `datasource-type=mysql [375]`

  Database type

- • **--replication-user=tungsten [397]**
  
  `replication-user=tungsten [397]`

  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 [396]**
  
  `replication-password=secret [396]`

  The password to be used when connecting to the database using the corresponding `--replication-user [397]`.

- • **--replication-host=remotedbhost [396]**
  
  `replication-host=remotedbhost [396]`

  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 slave (relay) connection.

  `replication-host [396]` should only be added to the above configuration if the target MySQL Database is on a different host to the applier installation.

### 5.6.3.2. Amazon RDS and Amazon Aurora Targets

- Amazon RDS and Amazon Aurora Targets

  Configure the installation using `tpm`:

  ```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
  ```
### Deploying Appliers

```bash
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>
vim /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

[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**
  
  reset

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

- **--install-directory=/opt/continuent**

  install-directory=/opt/continuent

  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**

  user=tungsten

  System User

- **--mysql-allow-intensive-checks=true**

  mysql-allow-intensive-checks=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.

- **--profile-script=~/.bash_profile**

  profile-script=~/.bash_profile

  Append commands to include env.sh in this profile script

- **--skip-validation-check=InstallerMasterSlaveCheck**

  skip-validation-check=InstallerMasterSlaveCheck
Deploying Appliances

The \texttt{--skip-validation-check} disables a given validation check. If any validation check fails, the installation, validation or configuration will automatically stop.

\textbf{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 \texttt{MySQLDefaultTableTypeCheck}, and could be ignored using \texttt{--skip-validation-check=MySQLDefaultTableTypeCheck}.

Setting both \texttt{--skip-validation-check} and \texttt{--enable-validation-check} is equivalent to explicitly disabling the specified check.

- \texttt{--skip-validation-check=MySQLPermissionsCheck}
- \texttt{skip-validation-check=MySQLPermissionsCheck}

The \texttt{--skip-validation-check} disables a given validation check. If any validation check fails, the installation, validation or configuration will automatically stop.

\textbf{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 \texttt{MySQLDefaultTableTypeCheck}, and could be ignored using \texttt{--skip-validation-check=MySQLDefaultTableTypeCheck}.

Setting both \texttt{--skip-validation-check} and \texttt{--enable-validation-check} is equivalent to explicitly disabling the specified check.

- \texttt{--skip-validation-check=MySQLBinaryLogsEnabledCheck}
- \texttt{skip-validation-check=MySQLBinaryLogsEnabledCheck}

The \texttt{--skip-validation-check} disables a given validation check. If any validation check fails, the installation, validation or configuration will automatically stop.

\textbf{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 \texttt{MySQLDefaultTableTypeCheck}, and could be ignored using \texttt{--skip-validation-check=MySQLDefaultTableTypeCheck}.

Setting both \texttt{--skip-validation-check} and \texttt{--enable-validation-check} is equivalent to explicitly disabling the specified check.

- \texttt{--skip-validation-check=MySQLMyISAMCheck}
- \texttt{skip-validation-check=MySQLMyISAMCheck}
Deploying Appliers

The `--skip-validation-check` 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`, and could be ignored using `--skip-validation-check=MySQLDefaultTableTypeCheck`.

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

**Configuration group** alpha

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

- **--master=sourcehost**
  
  The hostname of the master (extractor) within the current service. If the current host does not match this specification, then the deployment will by default be configured as a master/extractor.

- **--members=localhost,sourcehost**
  
  Hostnames for the dataservice members

- **--datasource-type=mysql**
  
  Database type

- **--datasource-mysql-conf=/dev/null**
  
  Database configuration
Deploying Appliers

MySQL config file

- --replication-user=rdsuser
- replication-user=rdsuser

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
- replication-password=secret

The password to be used when connecting to the database using the corresponding --replication-user.

- --privileged-slave=false
- privileged-slave=false

Does the login for the slave database service have superuser privileges

- --replication-host=rds-endpoint-url
- replication-host=rds-endpoint-url

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 slave [relay] connection.

- --service-type=remote
- service-type=remote

What is the replication service type?

5.6.4. 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 trepcfl. On the master:

```
shell> trepctl status
Processing status command...
NAME VALUE
appliedLastEventId: mysql-bin.000043:0000000000000291:84
appliedLastSeqno: 2320
appliedLatency: 0.733
channel: 1
clusterName: alpha
currentEventId: mysql-bin.000043:0000000000000291
currentTimeMillis: 1387544952494
dataServerHost: host1
extensions: host1
latcntEpochNumber: 60
masterConnectUri: thl://localhost:128
masterListenUri: thl://host1:2112/
maxStoredSeqno: 2320
minStoredSeqno: 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
```
5.7. Deploying the Oracle Applier

<table>
<thead>
<tr>
<th>Replication Operation Support</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Statements Replicated</td>
<td>No</td>
</tr>
<tr>
<td>Rows Replicated</td>
<td>Yes</td>
</tr>
<tr>
<td>Schema Replicated</td>
<td>No</td>
</tr>
<tr>
<td>ddlscan Supported</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Tungsten Clustering 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.
Deploying Appliers

Replication in these configurations operates using two separate replicators:

- Replicator on the master extracts the information from the source database into THL.
- Replicator on the slave reads the information from the remote replicator as THL, and applies that to the target database.

5.7.1. 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 5.7.1, “Additional Prerequisites for Oracle Targets” then finally follow the appropriate steps for the required extractor topology outlined in Chapter 3, *Deploying MySQL Extractors* for MySQL extraction, or Chapter 4, *Deploying Oracle Extractors* for Oracle CDC extraction before configuring the Oracle applier.

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 4.1, “Data Type differences when replicating data from MySQL to Oracle”.

Table 5.4. Data Type differences when replicating data from MySQL to Oracle

<table>
<thead>
<tr>
<th>MySQL Datatype</th>
<th>Oracle Datatype</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT</td>
<td>NUMBER(10, 0)</td>
<td></td>
</tr>
<tr>
<td>BIGINT</td>
<td>NUMBER(19, 0)</td>
<td></td>
</tr>
<tr>
<td>TINYINT</td>
<td>NUMBER(3, 0)</td>
<td></td>
</tr>
<tr>
<td>SMALLINT</td>
<td>NUMBER(5, 0)</td>
<td></td>
</tr>
<tr>
<td>MEDIUMINT</td>
<td>NUMBER(7, 0)</td>
<td></td>
</tr>
<tr>
<td>DECIMAL(x,y)</td>
<td>NUMBER(x, y)</td>
<td></td>
</tr>
<tr>
<td>FLOAT</td>
<td>FLOAT</td>
<td></td>
</tr>
<tr>
<td>CHAR(n)</td>
<td>CHAR(n)</td>
<td></td>
</tr>
<tr>
<td>VARCHAR(n)</td>
<td>VARCHAR2(n)</td>
<td>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</td>
</tr>
<tr>
<td>DATE</td>
<td>DATE</td>
<td></td>
</tr>
<tr>
<td>DATETIME</td>
<td>DATE</td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>DATE</td>
<td></td>
</tr>
</tbody>
</table>
MySQL Datatype | Oracle Datatype | Notes
---|---|---
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.
BLOB | BLOB |
ENUM(...) | VARCHAR(255) | Use the EnumToScript filter
SET(...) | VARCHAR(255) | Use the SetToScript 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 data server 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 Clustering 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 Clustering 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.

5.7.1.1. 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-5.4.1-41/tungsten_replicator/lib`

5.7.1.2. Configure the Oracle database

Before installing replication, the Oracle target database must be configured:
Deploying Appliers

- 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 Clustering 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
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 (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
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
SQL> GRANT CONNECT TO tungsten_alpha;
SQL> GRANT DBA TO tungsten_alpha;
```

The user/password combination selected will be required when configuring the slave replication service.

### 5.7.1.3. Create the Destination Schema

On the host which has been already configured as the master, use **ddlscan** to extract the DDL for Oracle:

```shell
shell> cd tungsten-replicator-5.4.1-41
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
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
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.

### 5.7.2. Install Oracle Applier

The slave replicator will read the THL from the remote master and apply it into Oracle using a standard JDBC connection. The slave replicator needs to know the master hostname, and the datasource type.

1. Unpack the Tungsten Replicator distribution in staging directory:

   ```shell
   shell> tar zxf tungsten-replicator-5.4.1-41.tar.gz
   ```

2. Change into the staging directory:

   ```shell
   shell> cd tungsten-replicator-5.4.1-41
   ```

3. Obtain a copy of the Oracle JDBC driver and copy it into the **tungsten-replicator/lib** directory:

   ```shell
   shell> cp ojdbc7.jar ./tungsten-replicator/lib/
   ```

4. Configure the installation using **tpm**:

   **Show Staging**

   ```shell
   shell> ./tools/tpm configure defaults \
       --reset \
       --install-directory=/opt/continuent \
       --profile-script=~/.bash_profile \
       --skip-validation-check=InstallerMasterSlaveCheck
   ```

   **Show INI**

   ```shell
   shell> ./tools/tpm configure alpha \
       --master=sourcehost \
       --members=localhost \
   ```

142
Deploying Appliers

```
--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

[alpha]
master=sourcehost
members=localhost
datasource-type=oracle
datasource-oracle-service=ORCL
datasource-user=tungsten_alpha
datasource-password=secret
dsvc-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**
  - reset
  - For staging configurations, deletes all pre-existing configuration information between updating with the new configuration values.

- **--install-directory=/opt/continuent**
  - install-directory=/opt/continuent
  - 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**
  - profile-script=~/.bash_profile
  - Append commands to include env.sh in this profile script

- **--skip-validation-check=InstallerMasterSlaveCheck**
  - skip-validation-check=InstallerMasterSlaveCheck
  - The **--skip-validation-check** 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** [346], and could be ignored using **--skip-validation-check=MySQLDefaultTableTypeCheck** [335].

  Setting both **--skip-validation-check** [335] and **--enable-validation-check** [333] is equivalent to explicitly disabling the specified check.

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.

- \texttt{--master=sourcehost [386]}
  
  master=sourcehost [386]

  The hostname of the master (extractor) within the current service. If the current host does not match this specification, then the deployment will by default be configured as a master/extractor.

- \texttt{--members=localhost [386]}
  
  members=localhost [386]

  Hostnames for the dataservice members

- \texttt{--datasource-type=oracle [375]}
  
  datasource-type=oracle [375]

  Database type

- \texttt{--datasource-oracle-service=ORCL [374]}
  
  datasource-oracle-service=ORCL [374]

  Oracle Service

- \texttt{--datasource-user=tungsten_alpha [397]}
  
  datasource-user=tungsten_alpha [397]

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

- \texttt{--datasource-password=secret [396]}
  
  datasource-password=secret [396]

  The password to be used when connecting to the database using the corresponding \texttt{--replication-user [397]}. 

- \texttt{--svc-applier-filters=dropstatementdata [399]}
  
  svc-applier-filters=dropstatementdata [399]

  Replication service applier filters

- \texttt{--replication-host [396]}
  
  \texttt{replication-host [396]} should be added to the above configuration if the target Oracle Database is on a different host to the applier installation

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

\begin{verbatim}
shell> ./tools/tpm install
\end{verbatim}

If the installation process fails, check the output of the \texttt{/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 master replicator.

The status of the replicator can be checked and monitored by using the \texttt{trepctl} command.

\section*{5.8. 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 or Oracle 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 Tungsten Clustering 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 the s3cmd command. 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 5.9. Topologies: Replicating to Amazon Redshift

Setting up replication requires setting up both the master and slave components as two different configurations, one for MySQL (or Oracle) 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.

5.8.1. Redshift Replication Operation

The Redshift applier makes use of the JavaScript based batch loading system [see Section 6.5.4, "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 5.10, "Topologies: Redshift Replication Operation".
Different object types within the two systems are mapped as follows:

<table>
<thead>
<tr>
<th>MySQL</th>
<th>Redshift</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance</td>
<td>Database</td>
</tr>
<tr>
<td>Database</td>
<td>Schema</td>
</tr>
<tr>
<td>Table</td>
<td>Table</td>
</tr>
</tbody>
</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 10.4.8, "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 10.4.30, "PrimaryKey Filter" filter is used to extract primary key data from the source tables.

3. On the slave 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 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 \x01. This is supported by Hive as a native value separator.
- The content of the file consists of the full row data extracted from the master, plus metadata describing the operation for each row, the sequence number, and then the full row information.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Sequence No</th>
<th>Table-specific primary key</th>
<th>DateTime</th>
<th>Table-columns...</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPTYPE</td>
<td>seqno [503] that generated this row</td>
<td>PRIMARYKEY</td>
<td>DATETIME of source table commit</td>
<td></td>
</tr>
</tbody>
</table>

The operation field will match one of the following values

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Row is an INSERT of new data</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Row is DELETE of existing data</td>
<td></td>
</tr>
<tr>
<td>UI</td>
<td>Row is an UPDATE which caused INSERT of data</td>
<td></td>
</tr>
<tr>
<td>UD</td>
<td>Row is an UPDATE which caused DELETE of data</td>
<td></td>
</tr>
</tbody>
</table>

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.

5.8.2. 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 for MySQL extraction, or Chapter 4, Deploying Oracle Extractors for Oracle CDC extraction.

The following are required for replication to Amazon Redshift:

5.8.2.1. 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 5.8.2.2, “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 installed and configured. The s3cmd can be downloaded from s3cmd on s3tools.org.

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:

```
[default]
access_key = ACCESS_KEY
```

147
Deploying Appliers

- 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 Clustering 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.
- `awsSecretKey` — the IAM role configured to allow Redshift to interact with S3
- `awsS3Path` — the S3 secret key to access your S3 storage.
- `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.
- `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 5.8.5, “Keeping CDC Information”.

### 5.8.2.2. Configuring Identity Access Management within AWS

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

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

**Note**

All versions up to and including Tungsten Replicator v5.3, and version v6.0 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 v5.4 and v6.1 will 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:

**For use by s3cmd to load csv into 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 instruction on creating and managing IAM roles, review the AWS documentation

---

### 5.8.2.3. 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 source (master) 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:

   ```shell
   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
   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.
5.8.3. 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 6.2, “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 parameters, update the extractor and then install the applier

   • For Staging Install:

     ```
     shell> cd tungsten-replicator-5.4.1-41
     shell> ./tools/tpm configure alpha --enable-heterogeneous-service=true
     shell> ./tools/tpm update
     ```

   • For INI Installs: Add the following the `/etc/tungsten/tungsten.ini`

     ```ini
     [alpha]
     ...Existing Replicator Config...
     enable-heterogeneous-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-5.4.1-41.tar.gz
   ```

3. Change into the staging directory:

   ```
   shell> cd tungsten-replicator-5.4.1-41
   ```

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
   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-database=dev --batch-enabled=true --batch-load-template=redshift --svc-applier-block-commit-interval=10s --svc-applier-block-commit-size=5
   shell> vi /etc/tungsten/tungsten.ini
   ```

   ```ini
   [defaults]
   user=tungsten
   install-directory=/opt/continuent
   profile-script=~/.bash_profile
   
   [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
tsvc-applier-block-commit-interval=10s
tsvc-applier-block-commit-size=5

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` [397]
  
  reset [397]

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

- `--user=tungsten` [403]
  
  user=tungsten [403]

  System User

- `--install-directory=/opt/continuent` [382]
  
  install-directory=/opt/continuent [382]

  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` [394]
  
  profile-script=~/.bash_profile [394]

  Append commands to include env.sh in this profile script

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` [403]
  
  topology=master-slave [403]

  Replication topology for the dataservice. Valid values are star, cluster-slave, master-slave, fan-in, clustered, cluster-alias, all-masters, direct

- `--master=sourcehost` [386]
  
  master=sourcehost [386]

  The hostname of the master (extractor) within the current service. If the current host does not match this specification, then the deployment will by default be configured as a master/extractor.

- `--members=localhost` [386]
  
  members=localhost [386]

  Hostnames for the dataservice members

- `--datasource-type=redshift` [375]
  
  datasource-type=redshift [375]

  Database type

- `--replication-host=redshift.us-east-1.redshift.amazonaws.com` [396]
  
  replication-host=redshift.us-east-1.redshift.amazonaws.com [396]
Deploying Appliers

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 slave (relay) connection.

- `--replication-user=awsRedshiftUser`

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`

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

- `--redshift-dbname=dev`

Name of the Redshift database to replicate into

- `--batch-enabled=true`

Should the replicator service use a batch applier

- `--batch-load-template=redshift`

Value for the `loadBatchTemplate` property

- `--svc-applier-filters=dropstatementdata`

Replication service applier filters

- `--svc-applier-block-commit-interval=10s`

Minimum interval between commits

- `--svc-applier-block-commit-size=5`

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 Elasticsearch applier slave host(s) only:

```shell
mkdir -p /opt/continuent/share/
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. 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:
Deploying Appliers

```shell
cp s3-config-service-name.json /opt/continuent/share/
```

Now the services can be started:

```shell
tpm start alpha
```

Once the service is configured and running, the service can be monitored as normal using the `trepctl` command. See Section 5.8.6, "Management and Monitoring of Amazon Redshift Deployments" for more information.

### 5.8.4. 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
   ```

### 5.8.5. 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 `ddiscan`:

   ```shell
   shell> ddiscan -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 `ddiscan` to define the CDC tables:

   ```json
   {
   "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.

### 5.8.6. Management and Monitoring of Amazon Redshift Deployments

Monitoring a Amazon Redshift replication scenario requires checking the status of both the master - extracting data from MySQL - and the slave which retrieves the remote THL information and applies it to Amazon Redshift.

```shell
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 5.4.1 build 41
Finished status command...
```
Deploying Appliers

On the slave, 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
clusternode : alpha
currentEventId : NONE
currentTimeMillis : 1407336316454
databasename : redshift.us-east-1.redshift.amazonaws.com
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
sourceId : redshift.us-east-1.redshift.amazonaws.com
state : ONLINE
timeInStateSeconds : 2.28
transitioningTo : 
uptimeSeconds : 524104.751
useSSLConnection : false
version : Tungsten Replicator 5.4.1 build 41
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 11.1, “Block Commit” for more information on these parameters.

5.9. 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 Clustering 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.
Deploying Appliers

5.9.1. 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 for MySQL extraction, or Chapter 4, Deploying Oracle Extractors for Oracle CDC extraction.

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 -U dbadmin -w secret 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
         \q or terminate with semicolon to execute query
         \q to quit
  bigdata=> create schema tungsten_alpha;
  ```

  The schema will be used only by Tungsten Clustering 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 Clustering lib directory.

  ```
  shell> cp vertica-jdbc-7.1.2-0.jar tungsten-replicator-5.4.1-41/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 Clustering deployment directory, the ddiscan 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:

```bash
shell> cd tungsten-replicator-5.4.1-41
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
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:

```bash
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
```

This generates the table definitions for live data. The process should be repeated to create the table definitions for the staging data by using the staging template:

```bash
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:

```bash
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 Clustering

### 5.9.2. Install Vertica Applier

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:

     ```bash
     shell> cd tungsten-replicator-5.4.1-41
     shell> ./tools/tpm configure alpha \
     --repl-svc-extractor-filters=colnames,pkey \
     --property=replicator.filter.pkey.addColumnsToDeletes=true \
     --property=replicator.filter.pkey.addPkeyToInserts=true \
     shell> ./tools/tpm update
     ```

   - For INI Installs: Add the following to the `/etc/tungsten/tungsten.ini`

     ```ini
     [alpha]
     ...Existing Replicator Config...
     repl-svc-extractor-filters=colnames,pkey
     property=replicator.filter.pkey.addColumnsToDeletes=true
     property=replicator.filter.pkey.addPkeyToInserts=true
     ```
2. The applier can now be configured.

Unpack the Tungsten Replicator distribution in staging directory:

```
shell> tar zxf tungsten-replicator-5.4.1-41.tar.gz
```

3. Change into the staging directory:

```
shell> cd tungsten-replicator-5.4.1-41
```

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 Clustering lib directory.

```
shell> cp vertica-jdbc-7.1.2-0.jar tungsten-replicator-5.4.1-41/tungsten-replicator/lib/
```

5. Configure the installation using tpm:

Show Staging

```
shell> ./tools/tpm configure defaults --reset --user=tungsten --install-directory=/opt/continuent --profile-script=~/.bash_profile --skip-validation-check=HostsFileCheck --skip-validation-check=InstallerMasterSlaveCheck
```

Show INI

```
```

```
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

[alpha]
topology=master-slave
master=sourcehost
members=localhost
datasource-type=vertica
replication-host=verticahost
replication-user=dbadmin
replication-password=password
redshift-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 [397]**
  
  reset [397]

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

- **--user=tungsten [403]**
  
  user=tungsten [403]

  System User

- **--install-directory=/opt/continuent [382]**
  
  install-directory=/opt/continuent [382]

  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 [394]**
  
  profile-script=~/.bash_profile [394]

  Append commands to include env.sh in this profile script

- **--skip-validation-check=HostsFileCheck [335]**
  
  skip-validation-check=HostsFileCheck [335]

  The **--skip-validation-check [335]** 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 [346]**, and could be ignored using **--skip-validation-check=MySQLDefaultTableTypeCheck [335]**.

  Setting both **--skip-validation-check [335]** and **--enable-validation-check [333]** is equivalent to explicitly disabling the specified check.

- **--skip-validation-check=InstallerMasterSlaveCheck [335]**
  
  skip-validation-check=InstallerMasterSlaveCheck [335]

  The **--skip-validation-check [335]** 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 [346]**, and could be ignored using **--skip-validation-check=MySQLDefaultTableTypeCheck [335]**.
Deploying Appliers

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

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`
  
  Topology for the dataservice. Valid values are: star, cluster-slave, master-slave, fan-in, clustered, cluster-alias, all-masters, direct.

- `--master=sourcehost`
  
  The hostname of the master (extractor) within the current service. If the current host does not match this specification, then the deployment will by default be configured as a master/extractor.

- `--members=localhost`
  
  Hostnames for the dataservice members.

- `--datasource-type=vertica`
  
  Database type.

- `--replication-host=verticahost`
  
  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 slave (relay) connection.

- `--replication-user=dbadmin`
  
  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`
  
  The password to be used when connecting to the database using the corresponding `--replication-user`.

- `--redshift-dbname=dev`
  
  Name of the Redshift database to replicate into.

- `--batch-enabled=true`
  
  Should the replicator service use a batch applier.

Value for the `loadBatchTemplate` property
Deploying Appliers

- `--batch-load-language=js` [367]
  batch-load-language=js [367]
  Which script language to use for batch loading

- `--replication-port=5433` [396]
  replication-port=5433 [396]
  The network port used to connect to the database server. The default port used depends on the database being configured.

- `--svc-applier-filters=dropstatementdata` [399]
  svc-applier-filters=dropstatementdata [399]
  Replication service applier filters

- `--svc-applier-block-commit-interval=30s` [399]
  svc-applier-block-commit-interval=30s [399]
  Minimum interval between commits

- `--svc-applier-block-commit-size=25000` [399]
  svc-applier-block-commit-size=25000 [399]
  Applier block commit size [min 1]

- `--disable-relay-logs=true` [378]
  disable-relay-logs=true [378]
  Disable the use of relay-logs?

`replication-host` [396] should only be added to the above configuration if the target MySQL Database is on a different host to the appli-
er installation

6. 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 5.9.3, “Manage-
ment and Monitoring of Vertica Deployments” for more information.

5.9.3. Management and Monitoring of Vertica Deployments

Monitoring a Vertica replication scenario requires checking the status of both the master - extracting data from MySQL - and the slave 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
```
On the slave, 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/
masterListenerUri      : 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                : 10008
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 5.4.1 build 41
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.

### 5.9.4. 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:
Deploying Appliers

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 directory 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 ... csv_columns=tungsten_opcode,tungsten_seqno,tungsten_row_id,tungsten_commit_timestamp,nullschemaname,schemahash
```

Indicates the source THL has not been marked up correctly. Either the `colnames` filter has not been enabled, or the `--enable-batch-service` has not been configured 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 master and slave should be recreated by using `trepctl reset`.

- If you get an error similar to the following:

```
pendingExceptionMessage: CSV loading failed: schema=test table=mg 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: 
```

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 required in order to load into Vertica through the batch mechanism. The configuration should be updated, and then the THL on both the master and slave 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 ="files/msg/test/alpha/staging0/blog-stage_xxx_article CSV ="LOAD DATA ROW count does not match: sql=COPY blog.stage_xxx_article FROM '/tmp/staging/alpha/staging0/blog-stage_xxx_article.csv' DIRECT NULL 'null' DELIMITER ',' ENCLOSED BY '"' expected_copy_rows=3614 rows=2233 ; exceptions are in ="/tmp/tungsten_vertica_blog.article.exceptions (...)/tungsten-connector/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 newlines 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 5.9.1, “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 re-write 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
```
Chapter 6. Advanced Deployments

6.1. Deploying a Fan-In Topology

The fan-in topology is the logical opposite of a master/slave topology. In a fan-in topology, the data from two masters is combined together on one slave. 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 6.1. 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 ny_sales.

  Alternatively, a filter can be configured to rename the database sales dynamically to the corresponding location based tables. See Section 10.4.32, “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 masters. 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 master and replicated to the slave and cause the operations to be triggered multiple times. Tungsten Clustering 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 master or slave. For more information, see Section C.3.1, “Triggers”.

To create the configuration the masters 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

164
Advanced Deployments

```bash
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
start-and-report=true
```

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 [403]**
  topology=fan-in [403]
  Replication topology for the dataservice Valid values are star, cluster-slave, master-slave, fan-in, clustered, cluster-alias, all-masters, direct

- **--install-directory=/opt/continuent [382]**
  install-directory=/opt/continuent [382]
  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 [397]**
  replication-user=tungsten [397]
  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 [396]**
  replication-password=password [396]
  The password to be used when connecting to the database using the corresponding --replication-user [397].

- **--master=host1,host2 [386]**
  master=host1,host2 [386]
  The hostname of the master (extractor) within the current service. If the current host does not match this specification, then the deployment will by default be configured as a master/extractor.

- **--members=host1,host2,host3 [386]**
  members=host1,host2,host3 [386]
  Hostnames for the dataservice members

- **--master-services=alpha,beta [386]**
  master-services=alpha,beta [386]
  Data service names that should be used on each master

- **--start-and-report=true [399]**
  start-and-report=true [399]
  Start the services and report out the status after configuration

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.

6.1.1. 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 master 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
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 5.4.1 build 41
Finished status command...
```

The corresponding master 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;8
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
```
Advanced Deployments

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:

<table>
<thead>
<tr>
<th>Master Service</th>
<th>Master Host</th>
<th>Slave Host</th>
</tr>
</thead>
<tbody>
<tr>
<td>alpha</td>
<td>host1</td>
<td>host3</td>
</tr>
<tr>
<td>beta</td>
<td>host1</td>
<td>host3</td>
</tr>
</tbody>
</table>

The sequence numbers between host1 and host2 will not match, as they are two independent services.

For more information on using trepctl, see Section 8.18, “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.

6.2. 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 heterogeneous 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:

6.2.1. 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 Section 2.3, “Prepare Hosts” and Appendix B, Prerequisites for additional details.

- RMI network port used for communicating with the replicator service.
  Set through the \texttt{--rmi-port} parameter to \texttt{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 \texttt{--thl-port} parameter to \texttt{tpm}. The default THL port is 2112. This option is required for services operating as masters [extractors].

- Master THL port, i.e. the port from which a slave will read THL events from the master
  Set through the \texttt{--master-thl-port} parameter to \texttt{tpm}. When operating as a slave, the explicit THL port should be specified to ensure that you are connecting to the THL port correctly.

- Master hostname
  Set through the \texttt{--master-thl-host} parameter to \texttt{tpm}. This is optional if the master hostname has been configured correctly through the \texttt{--master} parameter.

- Installation directory used when the replicator is installed.
Advanced Deployments

Set through the `--install-directory` or `--install-directory` 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".

### 6.2.2. 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-5.4.1-41.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-5.4.1-41
   ```

4. Extractor reading from MySQL (Click link to switch examples between Staging Method or INI Method):

   ```
   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
   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
   ```

   ![Configuration Example](example.png)

   ![Configuration Group](group.png)

   ![Configuration Group defaults](default.png)
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**

  resets all pre-existing configuration information between updating with the new configuration values.

- **--install-directory=/opt/extractor**

  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**

  System User

- **--profile-script=~/.bash_profile**

  Append commands to include env.sh in this profile script

- **--mysql-allow-intensive-checks=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.

- **--disable-security-controls=true**

  Disables all forms of security, including SSL, TLS and authentication

- **--executable-prefix=ext**

  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 be accessible through a unique alias. For example, if the executable prefix is configured as east, then an alias for the installation to trepc1 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.

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**

  The hostname of the master (extractor) within the current service. If the current host does not match this specification, then the deployment will by default be configured as a master/extractor.

- **--members=offboardhost**

  Hostnames for the dataservice members

- **--enable-heterogeneous-service=true**
enable-heterogeneous-service=true

- On a Master
  - --mysql-use-bytes-for-string 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 Slave
  - --mysql-use-bytes-for-string is set to true.
  - pkey filter is enabled (q-to-dbms stage).

replication-port=3306

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

replication-user=tungsten_alpha

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

The password to be used when connecting to the database using the corresponding --replication-user.

datasource-mysql-conf=/etc/my.cnf

MySQL config file

svc-extractor-filters=colnames,pkey

Replication service extractor filters

mysql-enable-enumtostring=true

Enable a filter to convert ENUM values to strings

mysql-enable-settostring=true

Enable a filter to convert SET types to strings

mysql-use-bytes-for-string=false

Transfer strings as their byte representation?

This is a standard configuration using the default ports, with the directory /opt/extractor.
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

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
```

```ini
[defaults]
install-directory=/opt/applier
profile-script=~/.bash_profile
skip-validation-check=InstallerMasterSlaveCheck
executable-prefix=app

[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** [397]

  reset [397]

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

- **--install-directory=/opt/applier** [382]

  install-directory=/opt/applier [382]

  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** [394]

  profile-script=~/.bash_profile [394]

  Append commands to include env.sh in this profile script

- **--skip-validation-check=InstallerMasterSlaveCheck** [335]

  skip-validation-check=InstallerMasterSlaveCheck [335]

  The `--skip-validation-check` [335] 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` [346], and could be ignored using `--skip-validation-check=MySQLDefaultTableTypeCheck` [335].

Setting both `--skip-validation-check` [335] and `--enable-validation-check` [333] is equivalent to explicitly disabling the specified check.

- `--executable-prefix=app` [381]

  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 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.

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` [386]
  
  `master=localhost` [386]

  The hostname of the master (extractor) within the current service. If the current host does not match this specification, then the deployment will by default be configured as a master/extractor.

- `--members=localhost` [386]
  
  `members=localhost` [386]

  Hostnames for the dataservice members

- `--role=slave` [397]
  
  `role=slave` [397]

  What is the replication role for this service?

- `--datasource-type=mongodb` [375]
  
  `datasource-type=mongodb` [375]

  Database type

- `--replication-user=tungsten` [397]
  
  `replication-user=tungsten` [397]

  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` [396]
  
  `replication-password=secret` [396]

  The password to be used when connecting to the database using the corresponding `--replication-user` [397].
**Advanced Deployments**

- `rmi-port=10002 [397]`
  Replication RMI listen port
- `--master-thl-port=2112 [386]`
  Master THL Port
- `--master-thl-host=localhost [386]`
  Master THL Hostname
- `--thl-port=2113 [403]`
  Port to use for THL Operations

In this configuration, the master 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` 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 [381]` 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 [394]` 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 [381]` 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
```

<table>
<thead>
<tr>
<th>host</th>
<th>servicename</th>
<th>role</th>
<th>state</th>
<th>appliedlastseqno</th>
<th>appliedlatency</th>
</tr>
</thead>
</table>
6.2.3. Best Practices: Multiple Replicators

Follow the guidelines in Section 2.2, “Best Practices”.

6.3. 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 master of the target dataservice. By writing this way, changes are replicated to the master and slave 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 Clustering service. For more information on initially provisioning the data for this type of operation, see Migrating from MySQL Native Replication Using a New Service [in Tungsten Clustering [for MySQL] 5.3 Manual].

Figure 6.2. 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 master of the destination dataservice.

2. Create a new replicator that reads the binary logs directly from the external MySQL service through the master 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 and DEB package files”], or download the compressed tarball and unpack it on host1:

```
shell> cd /opt/replicator/software
shell> tar xzf tungsten-replicator-5.4.1-41.tar.gz
```

Change to the Tungsten Replicator staging directory:

```
shell> cd tungsten-replicator-5.4.1-41
```

Configure the replicator on host1.
Advanced Deployments

First we configure the defaults and a cluster alias that points to the masters and slaves within the current Tungsten Clustering 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

shell> ./tools/tpm configure beta
   --topology=direct
   --master=host1
   --direct-datasource-host=host3
   --thl-port=2113

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

### 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**
  *install-directory=/opt/replicator*
  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**
  *rmi-port=10002*
  Replication RMI listen port

- **--user=tungsten**
  *user=tungsten*
  System User

- **--replication-user=tungsten**
  *replication-user=tungsten*
  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**
  *replication-password=secret*
  The password to be used when connecting to the database using the corresponding `--replication-user`.
Advanced Deployments

- **--skip-validation-check=MySQLNoMySQLReplicationCheck** [335]

```
skip-validation-check=MySQLNoMySQLReplicationCheck
```

The `--skip-validation-check` [335] 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` [346], and could be ignored using `--skip-validation-check=MySQLDefaultTableTypeCheck` [335].

Setting both `--skip-validation-check` [335] and `--enable-validation-check` [333] is equivalent to explicitly disabling the specified check.

**Configuration group beta**

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

- **--topology=direct** [403]

```
topology=direct
```

Replication topology for the datasource Valid values are star, cluster-slave, master-slave, fan-in, clustered, cluster-alias, all-masters, direct

- **--master=host1** [386]

```
master=host1
```

The hostname of the master (extractor) within the current service. If the current host does not match this specification, then the deployment will by default be configured as a master/extractor.

- **--direct-datasource-host=host3** [377]

```
direct-datasource-host=host3
```

Database server hostname

- **--thl-port=2113** [403]

```
thl-port=2113
```

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;
+---------------------------+-----------------------------+---------------------------+
| File                      | Position                    |
+---------------------------+-----------------------------+---------------------------+
| mysql-bin.000026          | 1311                        |
```

176
Advanced Deployments

Use `tungsten_set_position` to update the replicator position to point to the master log position:

```bash
shell> /opt/replicator/scripts/tungsten_set_position \    
   --seqno=0 --epoch=0 --service=beta \    
   --source-id=host3 --event-id=mysql-bin.000026:1311
```

Now start the replicator:

```bash
shell> /opt/replicator/tungsten/tungsten-replicator/bin/replicator start
```

Replication status should be checked by explicitly using the servicename and/or RMI port:

```bash
shell> /opt/replicator/tungsten/tungsten-replicator/bin/trepctl -service beta status
```

6.4. Deploying Parallel Replication

Parallel apply is an important technique for achieving high speed replication and curing slave lag. It works by spreading updates to slaves 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 slave. In ideal cases throughput on slaves may improve by up to 5 times over single-threaded MySQL native replication.

It is worth noting that the only thing Tungsten parallelizes is applying transactions to slaves. All other operations in each replication service are single-threaded. For a summary of the performance gains see the following article.

6.4.1. Application Prerequisites for Parallel Replication

Parallel replication works best on workloads that meet the following criteria:

- 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.
Advanced Deployments

- 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 slave host(s) are capable and have free memory in the OS page cache.
- The host on which the slave 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 slave 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 typically will not show much improvement with parallel apply.

### 6.4.2. Enabling Parallel Apply During Install

Parallel apply is enabled using the `svc-parallelization-type` and `channels` options of `tpm`. The parallelization type defaults to `none` which is to say that parallel apply is disabled. You should set it to `disk` in the `alpha` configuration group. The `channels` option sets 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 slave 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` [397]
  
  reset [397]

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

- `--install-directory=/opt/continuent` [382]
  
  install-directory=/opt/continuent [382]
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 [403]`
  `user=tungsten [403]`

System User

- `--mysql-allow-intensive-checks=true [389]`
  `mysql-allow-intensive-checks=true [389]`

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 [394]`
  `profile-script=~/.bash_profile [394]`

Append commands to include env.sh in this profile script

- `--start-and-report=true [399]`
  `start-and-report=true [399]`

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 [386]`
  `master=sourcehost [386]`

The hostname of the master [extractor] within the current service. If the current host does not match this specification, then the deployment will by default be configured as a master/extractor.

- `--members=localhost,sourcehost [386]`
  `members=localhost,sourcehost [386]`

Hostnames for the dataservice members

- `--datasource-type=mysql [375]`
  `datasource-type=mysql [375]`

Database type

- `--replication-user=tungsten [397]`
  `replication-user=tungsten [397]`

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 [396]`
  `replication-password=secret [396]`

The password to be used when connecting to the database using the corresponding `--replication-user [397]`

- `--svc-parallelization-type=disk [400]`
  `svc-parallelization-type=disk [400]`

Method for implementing parallel apply

- `--channels=10 [367]`
Advanced Deployments

channels=10 [367]

Number of replication channels to use for services

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** [367] — Sets the replicator block commit size, which is the number of transactions to commit at once on slaves. Values up to 100 are normally fine.

- **native-slave.Takeover** [391] — 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 slave by looking at the "channels" property once the replicator restarts.

```
slave shell> trepctl -service alpha status| grep channels
channels               : 10
```

### Important

The channel count for a Master will ALWAYS be 1 because extraction is single-threaded:

```
master 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 slave would be: Connections = Channel_Count x Service_Count x 2, so for a 4-way Composite Multimaster 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
```

### 6.4.3. 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` [367] 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 slave 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.

Affect of Channels on Backups

If you back up a slave that operates with more than one channel, say 30, you can only restore that backup on another slave 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.

6.4.4. Parallel Replication and Offline Operation

6.4.4.1. 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 slave 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.

6.4.4.2. 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

```
shell> tpm query staging
    tungsten@db1:/opt/continuent/software/tungsten-clustering-6.0.5-41
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-clustering-6.0.5-41
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
```

Show INI
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-clustering-6.0.5-41
```

```
shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
```

```
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-6.0.5-41
```

```
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 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.

### 6.4.4.3. 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 slave 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.

### 6.4.5. Adjusting Parallel Replication After Installation

#### 6.4.5.1. 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**

```
shell> tpm query staging
```

```
tungsten@db1:/opt/continuent/software/tungsten-clustering-6.0.5-41
```

```
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-clustering-6.0.5-41
```

```
shell> ssh {STAGING_USER}@{STAGING_HOST}
```

```
shell> cd {STAGING_DIRECTORY}
```

```
shell> ./tools/tpm configure defaults \
`--svc-parallelization-type=disk` \
`--channels=10`
```

182
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”.

```ini
[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-clustering-6.9.5-41
echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-6.9.5-41
cd (STAGING_DIRECTORY)
./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 slave by looking at the "channels" property once the replicator restarts.

```shell
trepctl -service alpha status| grep channels
```

**Important**

The channel count for a Master will ALWAYS be 1 because extraction is single-threaded:

```shell
trepctl -service alpha status| grep channels
```

**Warning**

Enabling parallel apply will dramatically increase the number of connections to the database server.

Typically the calculation on a slave would be: Connections = Channel_Count x Service_Count x 2, so for a 4-way Composite Multimaster 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   |
+----------------------+-------+
```

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;
SHOW VARIABLES LIKE 'max_connections';
```

```
+-----------------+-------+
| Variable_name   | Value |
+-----------------+-------+
| max_connections | 512   |
+-----------------+-------+
```

183
6.4.5.2. 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 slave 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-clustering-6.0.5-41
```
```
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-clustering-6.0.5-41
```
```
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-clustering-6.0.5-41
```
```
shell> echo The staging DIRECTORY is `tpm query staging| cut -d: -f2`
The staging DIRECTORY is /opt/continuent/software/tungsten-clustering-6.0.5-41
```
```
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 slave 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 slaves 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 slaves becoming inconsistent or failing. The cure is usually full resynchronization, so it is best to avoid this if possible.
6.4.5.3. 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
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 slave 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

```shell
Show INI
```

Run the `tpm` command to update the software with the Staging-based configuration:

```shell
Run the tpm command to update the software with the Staging-based configuration:
shell> .tools/tpm configure alpha \
   --svc-parallelization-type=none \
   --channels=1
```

For information about making updates when using a Staging-method deployment, please see Section 9.3.7, “Configuration Changes from a Staging Directory”.

Verification

You can check the number of active channels on a slave by looking at the "channels" property once the replicator restarts.

```shell
Verification
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 slaves becoming inconsistent and requiring full resynchronization. You should only do such cleanup under direction from Continuent support.
6.4.5.4. 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

Run the `tpm` command to update the software with the Staging-based configuration:

Run the `tpm` command to update the software with the INI-based configuration:

For information about making updates when using a Staging method deployment, please see Section 9.3.7, “Configuration Changes from a Staging Directory”.

For information about making updates when using an INI file, please see Section 9.4.4, “Configuration Changes with an INI file”.

6.4.6. 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.

6.4.6.1. Useful Commands for Parallel Monitoring Replication

The replicator has several helpful commands for tracking replication performance:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>trepctl status</code></td>
<td>Shows basic variables including overall latency of slave and number of apply channels</td>
</tr>
<tr>
<td><code>trepctl status -name shards</code></td>
<td>Shows the number of transactions for each shard</td>
</tr>
<tr>
<td><code>trepctl status -name stores</code></td>
<td>Shows the configuration and internal counters for stores between tasks</td>
</tr>
</tbody>
</table>
### 6.4.6.2. Parallel Replication and Applied Latency On Slaves

The `trepctl status` `appliedLastSeqno` parameter shows the sequence number of the last transaction committed. Here is an example from a slave with 5 channels enabled.

```sh
trepctl status
```

#### NAME                     VALUE
---                     -----  
appliedLastEventId     : mysql-bin.088211:000000000000000;0
appliedLastSeqno       : 78191
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 slave 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:

```
Show Staging

Show INI

Run the `tpm` command to update the software with the Staging-based configuration:
```

run the `tpm` command to update the software with the INI-based configuration:
```
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 : 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.

### 6.4.6.3. 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...  
...  
...  
  appliedLastEventId : mysql-bin.000211:0000000020094766;8  
  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 master 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.

### 6.4.6.4. 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 a 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)...  
...  
...  
  serializationCount : 26  
Finished status command...  
```

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.
### 6.4.6.5. 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 Staging
```

Show INI

```
Show INI
```

Run the `tpm` command to update the software with the Staging-based configuration:

```
Run the `tpm` command to update the software with the Staging-based configuration:
```

```
Run the `tpm` command to update the software with the INI-based configuration:
```

Run the `tpm` command to update the software with the INI-based configuration:

```
Run the `tpm` command to update the software with the INI-based configuration:
```

For information about making updates when using a Staging-method deployment, please see Section 9.3.7, "Configuration Changes from a Staging Directory".

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:

```
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:
```

### 6.4.6.6. 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.

```
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.
```
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> trepcctl status -name shards
Processing status command (shards)...
NAME    VALUE
----    -----            appliedLastEventId: mysql-bin.000211:800000000095529;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.000211:000000000095529;0
- SOURCEID = logos1
- METADATA = [mysql_server_id=1;service=percona;shard=#UNKNOWN]
  - foreign_key_checks = 1, unique_checks = 1, sql_mode = '', character_set_client = 8,
    collation_connection = 8, collation_server = 33]
- 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> trepcctl status -name tasks
Processing status command (tasks)...
NAME    VALUE
----    -----            appliedLastEventId: mysql-bin.000211:800000000095529;0
        appliedLastSeqno : 78023
        appliedLatency : 0.248
        applyTime : 0.003
        averageBlockSize : 2.520
        cancelled : false
        currentLastEventId: mysql-bin.000211:800000000095529;0
        currentLastFragno : 0
        currentLastSeqno : 78023
        eventCount : 5302
        extractTime : 274.907
        filterTime : 0.0
        otherTime : 0.0
        stage : q-to-dbms
        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.
6.4.7. 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.

6.4.8. 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` 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**

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 \([\text{memory} \text{ option}]\)

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

Run the `tpm` command to update the software with the Staging-based configuration:

For information about making updates when using a Staging-method deployment, please see Section 9.3.7, “Configuration Changes from a Staging Directory”.

Run the `tpm` command to update the software with the INI-based configuration:

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` 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.
6.5. Batch Loading for Data Warehouses

Tungsten Replicator normally applies SQL changes to slaves 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 GreenPlum 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.

6.5.1. How It Works

The batch applier loads data into the slave 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.

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>opcode</td>
<td>A transaction code that has the value &quot;I&quot; for insert and &quot;D&quot; for delete. Other types are available.</td>
</tr>
<tr>
<td>seqno</td>
<td>The Tungsten transaction sequence number</td>
</tr>
<tr>
<td>row_id</td>
<td>A line number that starts with 1 and increments by 1 for each new row</td>
</tr>
<tr>
<td>timestamp</td>
<td>The commit timestamp, i.e. the origin timestamp of the committed statement that generated the row information.</td>
</tr>
</tbody>
</table>

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.

6.5.2. 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.

6.5.3. 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 master 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` option. Here’s a typical vertica applier configuration, taken from Section 5.9, “Deploying the Vertica Applier”:

Show Staging

Show INI

```shell
shell> ./tools/tpm configure defaults \
   --reset \n   --user=tungsten \n   --install-directory=/opt/continuent \n   --profile-script=~/.bash_profile \n   --skip-validation-check=HostsFileCheck \n   --skip-validation-check=InstallerMasterSlaveCheck

shell> ./tools/tpm configure alpha \
   --topology=master-slave \n   --master=sourcehost \n   --members=localhost \n   --datasource-type=vertica \n   --replication-host=verticahost \n   --replication-user=dbadmin \n   --replication-password=password \n   --redshift-dbname=dev \n   --batch-enabled=true \n   --batch-load-template=vertica6 \n   --batch-load-language=js \n   --replication-port=5433 \n   --svc-applier-filters=dropstatementdata \n   --svc-applier-block-commit-interval=30s \n   --svc-applier-block-commit-size=25000 \n   --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

[alpha]
topology=master-slave
master=sourcehost
members=localhost
datasource-type=vertica
replication-host=verticahost
replication-user=dbadmin
replication-password=password
redshift-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
```

6.5.4. 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 6.3, “Batchloading: JavaScript”.

Figure 6.3. 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

### 6.5.4.1. 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.
Advanced Deployments

- 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.

6.5.5. Staging Tables

Staging tables are intermediate tables that help with data loading. There are different usage patterns for staging tables.

6.5.5.1. 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.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stageColumnPrefix</td>
<td>Prefix for seqno, row_id, and opcode columns generated by Tungsten</td>
</tr>
<tr>
<td>stageTablePrefix</td>
<td>Prefix for stage table name</td>
</tr>
<tr>
<td>stageSchemaPrefix</td>
<td>Prefix for the schema in which the stage tables reside</td>
</tr>
</tbody>
</table>

These values are set in the static properties file that defines the replication service. They can be set at install time using `--property` options. The following example shows typical values from a service properties file.

```java
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.

6.5.5.2. 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:

```sql
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.

```sql
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.

6.5.5.3. 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 following example:

```sql
CREATE TABLE `stage_xxx_croc_mydata` ( 
  `id` int(11) NOT NULL,
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

### 6.5.5.4. 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.

### 6.5.6. 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
  ```

### 6.5.7. 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 6.1. Continuent Tungsten Directory Structure

<table>
<thead>
<tr>
<th>Format</th>
<th>Field Separator</th>
<th>Record Separator</th>
<th>Escape Sequence</th>
<th>Escaped Characters</th>
<th>Null Policy</th>
<th>Null Value</th>
<th>Show Headers</th>
<th>Use Quotes</th>
<th>Quote String</th>
<th>Suppressed Characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>hive</td>
<td>\u0001</td>
<td>\n</td>
<td>|</td>
<td>\u0001|</td>
<td>Use Null Value</td>
<td>\N</td>
<td>false</td>
<td>false</td>
<td></td>
<td>\n\r</td>
</tr>
<tr>
<td>mysql</td>
<td>,</td>
<td>\n</td>
<td>|</td>
<td>|</td>
<td>Use Null Value</td>
<td>\N</td>
<td>false</td>
<td>true</td>
<td>&quot;</td>
<td></td>
</tr>
<tr>
<td>oracle</td>
<td>,</td>
<td>\n</td>
<td>|</td>
<td>|</td>
<td>Use Null Value</td>
<td>\N</td>
<td>false</td>
<td>true</td>
<td>&quot;</td>
<td></td>
</tr>
<tr>
<td>vertica</td>
<td>,</td>
<td>\n</td>
<td>|</td>
<td>|</td>
<td>Skip Value</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>&quot;</td>
<td>\n</td>
</tr>
<tr>
<td>redshift</td>
<td>,</td>
<td>\n</td>
<td>&quot;</td>
<td></td>
<td>Skip Value</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>&quot;</td>
<td>\n</td>
</tr>
</tbody>
</table>

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`).
6.5.8. 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 6.5.9, “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` 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`).

6.5.9. 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 enabled, 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.

6.5.10. 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:

```sh
# 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
```
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.

### 6.5.11. 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:

```java
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.

### 6.6. Deployment Security

Tungsten Replicator supports SSL, TLS and certificates for both communication and authentication. This security is disabled by default and includes:
• 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.

• File permissions and access by all components.

If you are using a single staging directory to handle your complete installation, `tpm` will automatically create the necessary certificates for you. If you fit in the below categories, you will need to use manually generated certificates.

• Installing heterogeneous replication using independent configurations

• MSMM, Cluster-Slave replication or anything using multiple Continuent packages

• Installing from multiple Staging Directories

**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.

### 6.6.1. Enabling Security

By default, security is disabled for the entire installation.

Security can be enabled by using the `--disable-security-controls=false` option to the `tpm` command:

```
• --disable-security-controls=false
```

This has the same effect as adding `--file-protection-level=0027`, `--rmi-ssl=true`, `--thl-ssl=true`, and `--rmi-authentication=true`.

**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 6.6.1.2, “Enabling Security using the INI Method” for details.

### 6.6.1.1. Enabling Security using the Staging Method

Security can be enabled either during initial installation or via an update.

For many reasons, it is much easier to enable SSL at install time. Both procedures follow below.

Enabling During Install

Security can be enabled at install time by using the `--disable-security-controls=false` option to the `tpm configure` command.

```
shell> tools/tpm configure defaults --disable-security-controls=false \ 
[the rest of the configuration options...]
shell> tools/tpm install
```

**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

Security can be enabled after install time by using the `--disable-security-controls=false` option to the `tpm configure` command followed by a special invocation of the `tpm update` command.

```
shell> tools/tpm configure defaults --disable-security-controls=false
shell> tools/tpm update --replace-jgroups-certificate --replace-tls-certificate --replace-release
```
Warning

This update will force replicator processes to be restarted.

6.6.1.2. Enabling Security using the INI Method

Security can be enabled either during initial installation or via an update.
For many reasons, it is much easier to enable SSL at install time. Both procedures follow below.

Enabling During Install

- First, configure the `tungsten.ini` file as follows:

  ```
  disable-security-controls=false
  start-and-report=false
  ```

- Next, do the fresh install on each node, which will generate new, different certificates on every node.

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

- You must then select one of the nodes and copy that node’s certificate files to all other nodes.
  For example, to seed a 6-node composite cluster, login to db1 and copy both the main and backup files to the other five nodes:

  ```
  shell> for i in `seq 2 6`; do scp /opt/continuent/share/[jpt]* db$i:/opt/continuent/share/; done
  ```

- On all nodes:

  ```
  shell> startall
  ```

Enabling Post-Installation

Security can be enabled after install time by updating the `tungsten.ini` file, followed by a special invocation of the `tpm update` command on all nodes.

- First, configure the `tungsten.ini` file as follows:

  ```
  disable-security-controls=false
  start-and-report=false
  ```

- 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:

  For example, to seed a 6-node composite cluster, login to db1 and copy both the main and backup files to the other five nodes:

  ```
  shell> for i in `seq 2 6`; do scp /opt/continuent/share/[jpt]* db$i:/opt/continuent/share/; done
  ```

- On all nodes:

  ```
  shell> startall
  ```

6.6.2. Disabling Security

There may be situations where security must be disabled for the entire installation.

Security can be disabled in the following ways during configuration with `tpm`:
• --disable-security-controls=true

This has the same effect as adding --file-protection-level=none, --rmi-ssl=false, --thl-ssl=false, --rmi-authentication=false.

• --file-protection-level=none

Disables file level protection, including ownership and file mode settings.

• --rmi-ssl=false

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.

• --rmi-authentication=false

Disables the use of authentication when accessing and controlling services.

• --thl-ssl=false

Disables the use of SSL/TLS for THL transmission between replicators.

6.6.3. 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 the following steps:

• Generating a TLS Certificate

Run this command to create the keystore in /etc/tungsten. You may use your own location, but the values for -storepass and -keypass must match.

```
keytool -genkey -alias tls \
  -validity 365 \ 
  -keyalg RSA -keystore /etc/tungsten/tls.jks \ 
  -dname "cn=Continuent, ou=IT, o=Continuent, c=US" \ 
  -storepass mykeystorepass -keypass mykeystorepass
```

6.6.4. Installing from a Staging Host with Manually Generated Certificates

Follow the steps in Section 6.6.3, “Creating Suitable Certificates” to create the TLS certificate.

Update your configuration to specify these certificates and the keystore password:

```
shell> tools/tpm configure SERVICE \ 
  --java-tls-keystore-path=/etc/tungsten/tls.jks \ 
  --java-keystore-password=mykeystorepass
```

6.6.5. Installing via INI File with Manually Generated Certificates

Follow the steps in Section 6.6.3, “Creating Suitable Certificates” to create the TLS certificate.

• Transfer the generated certificates to the same path on all hosts.

• Update your configuration to specify these certificates and the keystore password:

```
java-tls-keystore-path=/etc/tungsten/tls.jks
java-keystore-password=mykeystorepass
```

6.6.6. 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 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
```

If you do not meet these requirements, generate a new certificate and update it through the tpm command.
6.6.7. Removing TLS Encryption from a Staging Directory

Using the `tpm configure` command, the general Continuent service encryption can be easily removed.

```shell
shell> tpm configure SERVICE
   --thl-ssl=false
   --rmi-ssl=false
   --rmi-authentication=false
```

Then perform an update and replace the entire release directory:

```shell
shell> tpm update --replace-release
```

6.6.8. Handling Database level Security

If you choose to enable database level SSL within your MySQL installation, there are a number of additional steps required to allow the Replicators 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
- 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 at the replicator level, then you should have the following parameter enabled within your installation:

   ```
   disable-security-controls=false
   ```

   As a result, you should have a number of files within `/opt/continuent/share`

   ```shell
   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
   ```

   If this is the case, skip the next step and move onto step 3

2. If you do not have SSL enabled at the replicator level and you require this, then follow the steps in Section 6.6.1, “Enabling Security” first

   If you do not require SSL at the replicator level, 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
   ```

3. Next, add the following parameters to your installation, but do not run `tpm update` yet:

   ```
   property=replicator.global.db.sslEnabled=true
   property=replicator.global.db.sslOptions=useSSL=true
   ```

4. 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

5. We 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 replicator SSL already enabled
6. Next, import the client certificate into the truststore

If replicator SSL already enabled

shell> keytool -import -alias mysql -trustcacerts -file /home/tungsten/ca.pem 
-keystore /opt/continuent/share/tungsten_truststore.ts

If replicator SSL not enabled

shell> keytool -import -alias mysql -trustcacerts -file /home/tungsten/ca.pem 
-keystore /home/tungsten/tungsten_truststore.ts

When prompted for a password, enter tungsten

7. 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

6.6.9. Creating the Truststore and Keystore

The SSL configuration works through two separate files that define the server and client side of the encryption configuration. Because individual hosts within a Tungsten Clustering configuration are both servers [when acting as a master, or when providing status information], and clients [when reading remote THL and managing nodes remotely], both the server and client side of the configuration must be configured.  

Configuration for all systems relies on two files, the truststore, which contains the server certificate information [the certificates it will accept from clients], and the keystore, which manages the client certificate information [the certificates that will be provided to servers]. The truststore and keystore hold SSL certificate information, and are password protected.

The keystore and truststore operate by holding one or more certificates that will be used for encrypting communication. The following certificate options are available:

- Create your own server and client certificates
- Create your own server certificates, get the server certificate signed by a Certificate Authority (CA), and use a corresponding signed client certificate
- Use a server and client certificate already signed by a CA. Care should be taken with these certificates, as they are associated with specific domains and/or hosts, and may cause problems in a dynamic environment.

In a multi-node environment such as Tungsten Clustering, all the hosts in the dataservice can use the same keystore and truststore certificates. The tpm command will distribute these files along with the configuration when a new installation is deployed, or when updating an existing deployment.

6.6.9.1. Creating Your Own Client and Server Certificates

Because the client and server components of the Tungsten Clustering configuration are the same, the same certificate can be used and add to both the keystore and truststore files.

The process is as follows:

1. Create the keystore and generate a certificate
2. Export the certificate
3. Import the certificate to the truststore

To start, use the supplied keytool to create a keystore and populate it with a certificate. The process asks for certain information. The alias is the name to use for the server and can be any identifier. When asked for the first and last name, use localhost, as this is used as the server identifier for the certificate. The other information should be entered accordingly.
Keystores (and truststores) also have their own passwords that are used to protect the store from updating the certificates. The password must be known as it is required in the configuration so that Tungsten Clustering can open the keystore and read the contents.

```
shell> keytool -genkey -alias replserver -keyalg RSA -keystore keystore.jks
Enter keystore password:
Re-enter new password:
What is your first and last name?
[Unknown]: localhost
What is the name of your organizational unit?
[Unknown]: My OU
What is the name of your organization?
[Unknown]: Continuent
What is the name of your City or Locality?
[Unknown]: Mountain View
What is the name of your State or Province?
[Unknown]: CA
What is the two-letter country code for this unit?
[Unknown]: US
Is CN=My Name, OU=My OU, O=Continuent, L=Mountain View, ST=CA, C=US correct?
[no]: yes
Enter key password for <any>
(RETURN if same as keystore password):
```

The above process has created the keystore and the 'server' certificate, stored in the file keystore.jks.

Alternatively, you can create a new certificate in a keystore non-interactively by specifying the passwords and certificate contents on the command-line:

```
shell> keytool -genkey -alias replserver \
-keyalg RSA -keystore keystore.jks \
-dname "cn=localhost, ou=IT, o=Continuent, c=US" \
-storepass password -keypass password
```

Now you need to export the certificate so that it can be added to the truststore as the trusted certificate:

```
shell> keytool -export -alias replserver -file client.cer -keystore keystore.jks
Enter keystore password:
Certificate stored in file <client.cer>
```

This has created a certificate file in client.cer that can now be used to populate your truststore. When added the certificate to the truststore, it must be identified as a trusted certificate to be valid. The password for the truststore must be provided. It can be the same, or different, to the one for the keystore, but must be known so that it can be added to the Tungsten Clustering configuration.

```
shell> keytool -import -v -trustcacerts -alias replserver -file client.cer -keystore truststore.ts
Enter keystore password:
Re-enter new password:
```

This has created the truststore file, truststore.ts.

A non-interactive version is available by using the -noprompt option and supplying the truststore name:

```
shell> keytool -import -trustcacerts -alias replserver -file client.cer \
-keystore truststore.ts -storepass password -noprompt
```

The two files, the keystore [keystore.jks], and truststore [truststore.ts], along with their corresponding passwords can be now be used with the tpm command.
6.6.9.2. Creating a Custom Certificate and Getting it Signed

You can create your own certificate and get it signed by an authority such as VeriSign or Thawte. To do this, the certificate must be created first, then you create a certificate signing request, send this to your signing authority, and then import the signed certificate and the certificate authority certificate into your keystore and truststore.

Create the certificate:

```shell
keytool -genkey -alias replserver -keyalg RSA -keystore keystore.jks
```

Enter keystore password:

Re-enter new password:

What is your first and last name?

What is the name of your organizational unit?

What is the name of your organization?

What is the name of your City or Locality?

What is the name of your State or Province?

What is the two-letter country code for this unit?

Is CN=My Name, OU=My OU, O=Continuent, L=Mountain View, ST=CA, C=US correct?

Enter key password for <any>

Create a new signing request the certificate:

```shell
keytool -certreq -alias replserver -file certrequest.pem -keypass password -keystore keystore.jks -storepass password
```

This creates a certificate request, `certrequest.pem`. This must be sent the to the signing authority to be signed.

- **Official Signing**

Send the certificate file to your signing authority. They will send a signed certificate back, and also include a root CA and/or intermediary CA certificate. Both these and the signed certificate must be included in the keystore and truststore files.

First, import the returned signed certificate:

```shell
keytool -import -alias replserver -file signedcert.pem -keypass password -keystore keystore.jks -storepass password
```

Now install the root CA certificate:

```shell
keytool -import -alias careplserver -file cacert.pem -keypass password -keystore keystore.jks -storepass password
```

Note

If the import of your certificate with `keytool` fails, it may be due to an incompatibility with some versions of OpenSSL, which fail to create suitable certificates for third-party tools. In this case, see Section 6.6.9.4, "Converting SSL Certificates for keytool" for more information.

And an intermediary certificate if you were sent one:

```shell
keytool -import -alias interreplserver -file intercert.pem -keypass password -keystore keystore.jks -storepass password
```

Now export the signed certificate so that it can be added to the truststore. Although you can import the certificate supplied, by exporting the certificate in your keystore for inclusion into your truststore you can ensure that the two certificates will match:

```shell
keytool -export -alias replserver -file client.cer -keystore keystore.jks
```

Enter keystore password:

Certificate stored in file <client.cer>

The exported certificate and CA root and/or intermediary certificates must now be imported to the truststore:

```shell
keytool -import -trustcacerts -alias replserver -file client.cer -keystore truststore.ts -storepass password -noprompt
```

```shell
keytool -import -trustcacerts -alias careplserver -file cacert.pem -keystore truststore.ts -storepass password -noprompt
```

```shell
keytool -import -trustcacerts -alias interreplserver -file intercert.pem -keystore truststore.ts -storepass password -noprompt
```
• **Self-Signing**

If you have setup your own certificate authority, you can self-sign the request using `openssl`:

```
shell> openssl ca -in certrequest.pem -out certificate.pem
```

Convert the certificate to a plain PEM certificate:

```
shell> openssl x509 -in certificate.pem -out certificate.pem -outform PEM
```

Finally, for a self-signed certificate, you must combine the signed certificate with the CA certificate:

```
shell> cat certificate.pem cacert.pem > certfull.pem
```

This certificate can be imported into your keystore and truststore.

To import your signed certificate into your keystore:

```
shell> keytool -import -alias replserver -file certfull.pem -keypass password \
   -keystore keystore.jks -storepass password
```

Then export the certificate for use in your truststore:

```
shell> keytool -export -alias replserver -file client.cer -keystore keystore.jks \
   Enter keystore password:
   Certificate stored in file <client.cer>
```

The same certificate must also be exported and added to the truststore:

```
shell> keytool -import -trustcacerts -alias replserver -file client.cer \
   -keystore truststore.ts -storepass password -noprompt
```

This completes the setup of your truststore and keystore. The files created can be used in your `tpm` configuration.

### 6.6.9.3. Using an existing Certificate

If you have an existing certificate (for example with your MySQL, HTTP server or other configuration) that you want to use, you can import that certificate into your truststore and keystore. When using this method, you must import the signed certificate, and the certificate for the signing authority.

When importing the certificate into your keystore and truststore, the certificate supplied by the certificate authority can be used directly, but must be imported alongside the certificate authorities root and/or intermediary certificates. All the certificates must be imported for the SSL configuration to work.

The certificate should be in the PEM format if it is not already. You can convert to the PEM format by using the `openssl` tool:

```
shell> openssl x509 -in signedcert.crt -out certificate.pem -outform PEM
```

First, import the returned signed certificate:

```
shell> keytool -import -file certificate.pem -keypass password \
   -keystore keystore.jks -storepass password
```

#### Note

If the import of your certificate with `keytool` fails, it may be due to an incompatibility with some versions of OpenSSL, which fail to create suitable certificates for third-party tools. In this case, see Section 6.6.9.4, “Converting SSL Certificates for keytool” for more information.

Now install the root CA certificate:

```
shell> keytool -import -file cacert.pem -keypass password \
   -keystore keystore.jks -storepass password
```

And an intermediary certificate if you were sent one:

```
shell> keytool -import -file intercert.pem -keypass password \
   -keystore keystore.jks -storepass password
```

Now export the signed certificate so that it can be added to the truststore:

```
shell> keytool -export -alias replserver -file client.cer -keystore keystore.jks \
   Enter keystore password:
   Certificate stored in file <client.cer>
```

The exported certificate and CA root and/or intermediary certificates must now be imported to the truststore:
6.6.9.4. Converting SSL Certificates for keytool

Some versions of the openssl toolkit generate certificates which are incompatible with the certificate mechanisms of third-party tools, even though the certificates themselves work fine with OpenSSL tools and libraries. This is due to a bug which affected certain releases of openssl 1.0.0 and later and the X.509 certificates that are created.

This problem only affects self-generated and/or self-signed certificates generated using the openssl command. Officially signed certificates from Thawte, VeriSign, or others should be compatible with keytool without conversion.

To get round this issue, the keys can be converted to a different format, and then imported into a keystore and truststore for use with Tungsten Clustering.

To convert a certificate, use openssl to convert the X.509 into PKCS12 format. You will be prompted to enter a password for the generated file which is required in the next step:

```
shell> openssl pkcs12 -export -in client-cert.pem -inkey client-key.pem >client.p12
```

Enter Export Password:

Verifying - Enter Export Password:

To import the converted certificate into a keystore, specifying the destination keystore name, as well as the source PKCS12 password used in the previous step:

```
shell> keytool -importkeystore -srckeystore client.p12 -destkeystore keystore.jks -srcstoretype pkcs12
```

Enter destination keystore password:

Re-enter new password:

Enter source keystore password:

Entry for alias 1 successfully imported.

Import command completed: 1 entries successfully imported, 0 entries failed or cancelled

The same process can be used to import server certificates into truststore, by converting the server certificate and private key:

```
shell> openssl pkcs12 -export -in server-cert.pem -inkey server-key.pem >server.p12
```

Enter Export Password:

Verifying - Enter Export Password:

Then importing that into a truststore

```
shell> keytool -importkeystore -srckeystore server.p12 -destkeystore truststore.ts -srcstoretype pkcs12
```

Enter destination keystore password:

Re-enter new password:

Enter source keystore password:

Entry for alias 1 successfully imported.

Import command completed: 1 entries successfully imported, 0 entries failed or cancelled

For official CA certificates, the generated certificate information should be valid for importing using keytool, and this file should not need conversion.
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 slaves.

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:

```bash
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 Tungsten Clustering Home Directory

After installing Tungsten Clustering the home directory will be filled with a set of new directories. The home directory is specified by `--home-directory` or `--install-directory`. 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 Clustering.
- **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 Clustering tools.

7.2. Establishing the Shell Environment

The tools required to operate Tungsten Clustering 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` 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`.

```bash
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` and `--profile-script` 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` the `trepctl` command would be accessed as `mm_trepctl`.

7.3. Replicator Roles

Replicators can have one of two main roles, master or slave

- **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 receives data from a master and then applies that data to a target database or environment.

### 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:

```bash
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:

```bash
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 a slave, compared to the master from which it is processing information. In this example, the last sequence number and the latency between that sequence being processed on the master and applied to the slave is 17.66 seconds. You can compare this information to that provided by the master, either by logging into the master and running the same command, or by using the host command-line option:

```bash
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 master against the value on the slave, it is possible to determine that the slave and the master 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:

```bash
shell> trepctl status
Processing status command...
NAME                     VALUE
----                     -----  
appliedLastEventId     : mysql-bin.000064:0000000002757461:8  
appliedLastSeqno       : 212  
appliedLatency         : 263.43  
channels               : 1  
clusterName            : default  
currentEventId         : NONE  
currentTimeMillis      : 1365082088916  
dataServerHost         : host2  
extensions             :  
latestEpochNumber      : 0  
masterConnectURI       : thl://host1:2112/  
masterListenURI        : thl://host1:2112/  
maximumStoredSeqNo     : 724  
minimumStoredSeqNo     : 0  
offlineRequests        : NONE  
pendingError           : NONE  
pendingErrorCode       : NONE  
pendingErrorEventId    : NONE
```

210
Similar to the host specification, \texttt{trepctl} provides information for the default service. If you have installed multiple services, you must specify the service explicitly:

\begin{verbatim}
shell> trepctl -service servicename status
\end{verbatim}

If the service has been configured to operate on an alternative management port, this can be specified using the \texttt{-port} option. The default is to use port 10000.

The above command was executed on the slave host, \texttt{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, \texttt{mysql-bin.000064} is the filename of the binary log on the master. 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 slave database.

  When using parallel replication, this parameter returns the minimum applied sequence number among all the channels applying data.

- **appliedLatency**
  
  The \texttt{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, \texttt{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 a master, the value will be empty.

  On a slave, the URI of the master Tungsten Replicator from which the transaction data is being read from. The value supports multiple URIs (separated by comma) for topologies with multiple masters.

- **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 \texttt{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 a master, \texttt{pipelineSource} is the MySQL binary log. For a slave, \texttt{pipelineSource} is the THL of the master.

- **relativeLatency**
  
  The \texttt{relativeLatency} is the latency between now and timestamp of the last event written into the local THL. An increasing \texttt{relativeLatency} indicates that the replicator may have stalled and stopped applying changes to the dataserver.

- **state**
  
  The state parameter indicates the current state of the service. In this case, the state is \texttt{ONLINE}.
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**. 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 cluster is to compare the current sequence numbers and latencies for each slave compared to the master. For example:

```bash
shell> trepctl -h host2 status | grep applied
appliedLastEventId : mysql-bin.000076:0000000087725114:0
appliedLastSeqno   : 2445
appliedLatency     : 252.0
...
shell> trepctl -h 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 slave for performing updates.

### 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**
  
  The replicator service is starting up and reading the replicator properties configuration file.

- **OFFLINE:NORMAL**
  
  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**
  
  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**
  
  The replicator is currently reading provisioning information from the master database before entering the **ONLINE** state.

- **GOING-ONLINE:RESTORING**
  
  The replicator is preparing to go online and is currently restoring data from a backup.

- **GOING-ONLINE:SYNCHRONIZING**
  
  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 a slave has been switched online after maintenance, or in the event of a temporary network error where the slave has reconnected to the master.

- **ONLINE**
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 state.

- **GOING-OFFLINE**

  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** state. This state may also be seen in a node where auto-enable is disabled after a start or restart operation.

- **OFFLINE: NORMAL**

  This status will be seen on a **MASTER** replicator following the **ONLINE: DEGRADED** state and indicates that the replicator has completed reading all binlog 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** state indefinitely.

Maintenance on Tungsten Replicator or the dataserver must be performed while in the offline state. In the offline 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 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:

<table>
<thead>
<tr>
<th>Operation</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node operating normally</td>
<td><strong>ONLINE</strong> [212]</td>
</tr>
<tr>
<td>Administrator puts node into offline state</td>
<td><strong>GOING-OFFLINE</strong> [213]</td>
</tr>
<tr>
<td>Node is offline</td>
<td><strong>OFFLINE: NORMAL</strong> [212]</td>
</tr>
<tr>
<td>Administrator puts node into online state</td>
<td><strong>GOING-ONLINE: SYNCHRONIZING</strong> [212]</td>
</tr>
<tr>
<td>Node catches up with master</td>
<td><strong>ONLINE</strong> [212]</td>
</tr>
</tbody>
</table>

In the event of a failure, the sequence will trigger the node into the error state and then recovery into the online state:

<table>
<thead>
<tr>
<th>Operation</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node operating normally</td>
<td><strong>ONLINE</strong> [212]</td>
</tr>
<tr>
<td>Failure causes the node to go offline</td>
<td><strong>OFFLINE: ERROR</strong> [212]</td>
</tr>
<tr>
<td>Administrator fixes error and puts node into online state</td>
<td><strong>GOING-ONLINE: SYNCHRONIZING</strong> [212]</td>
</tr>
<tr>
<td>Node catches up with master</td>
<td><strong>ONLINE</strong> [212]</td>
</tr>
</tbody>
</table>

During an error state where a backup of the data is restored to a node in preparation of bringing the node back into operation:

<table>
<thead>
<tr>
<th>Operation</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node operating normally</td>
<td><strong>ONLINE</strong> [212]</td>
</tr>
<tr>
<td>Failure causes the node to go offline</td>
<td><strong>OFFLINE: ERROR</strong> [212]</td>
</tr>
<tr>
<td>Administrator restores node from backup data</td>
<td><strong>GOING-ONLINE: RESTORING</strong> [212]</td>
</tr>
<tr>
<td>Once restore is complete, node synchronizes with the master</td>
<td><strong>GOING-ONLINE: SYNCHRONIZING</strong> [212]</td>
</tr>
<tr>
<td>Node catches up with master</td>
<td><strong>ONLINE</strong> [212]</td>
</tr>
</tbody>
</table>

### 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** state if you are currently **ONLINE**:

```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: 21.489
uptimeSeconds: 935.072
```

To switch back to the **ONLINE** state:

```shell
trepctl online
```

When using replicator states in this manner, the replication between hosts is effectively paused. Any outstanding events from the master will be replicated to the slave with the replication continuing from the point where the node was switched to the **OFFLINE** state. The sequence number and latency will be reported accordingly, as seen in the example below where the node is significantly behind the master:

```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 master and slave dataserver can occur for a number of reasons, including:

- An update or insertion has occurred on the slave independently of the master. This situation can occur if updates are allowed on a slave that is acting as a read-only slave for scale out, or in the event of running management or administration scripts on the slave
- A switch or failover operation has lead to inconsistencies. This can happen if client applications are still writing to the slave or master 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 slave. The replication process stops applying changes to the slave when the first transaction fails to be applied to the slave. 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=128 fragno=0 message=java.sql.SQLException: Statement failed on slave but succeeded on master
pendingErrorEventId: mysql-bin.000012:0000000000012967:0
pendingErrorSeqno: 128
pendingExceptionMessage: java.sql.SQLException: Statement failed on slave but succeeded on master
...```

The `trepsvc.log` log file will also contain the error information about the failed statement. For example:

```
...```
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 slave to identify the faulty row. To correct this problem you will either need to skip the transaction or delete the offending row from the slave dataserver.

     The error will normally be identified due to the following error message when viewing the current replicator status, for example:

     ```
     > trepcntl 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
     ```

     The error can be generated when an insert or update has taken place on the slave rather than on the master.

     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:

     ```
     > 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:

     ```
     > 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 slave, 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 slave 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 master and slave 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 slave

  If the data on the slave is considered correct, or the data in both tables is the same or similar, the transaction from the master to the slave 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** 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 slave

  If the data on the master is considered canonical, then the data on the slave can be removed, and the replicator placed online.

  **Warning**

  Deleting data on the slave 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 slave

  If the transaction cannot be skipped, or the data safely deleted or modified, and only a single slave is affected, a backup of an existing, working, slave can be taken and restored to the broken slave.

  The `tungsten_provision_slave` command automates this process. See Section 7.6, “Provision or Reprovision a Slave” 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 slave from the master or another slave, 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 slave 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 Slave

The command performs three operations automatically:

1. Performs a backup of a remote slave
2. Copies the backup to the current host
3. Restores the backup

**Warning**

When using `tungsten_provision_slave` you must be logged in to the slave that has failed or that you want to reprovision. You cannot reprovision a slave remotely.

To use `tungsten_provision_slave`:

1. Log in to the failed slave.
2. Select the active slave within the dataservice that you want to use to reprovision the failed slave. You may use the master 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 `tungsten_provision_slave` specifying the source you have selected:

```
shell> tungsten_provision_slave --source=host2
```

The default backup service for the host will be used; `mysqldump` can be used by specifying the `--mysqldump` option.

`tungsten_provision_slave` handles the cluster status, backup, restore, and repositioning of the replication stream so that restored slave is ready to start operating again.

**Important**

When using a Multisite/Multimaster topology the additional replicator must be put offline before restoring data and put online after completion.

```
shell> mm_trepctl offline
shell> tungsten_provision_slave --source=host2
shell> mm_trepctl online
shell> mm_trepctl status
```
For more information on using `tungsten_provision_slave` see Section 8.21, “The `tungsten_provision_slave` 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.

Back 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 located 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. 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.

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 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` 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 Clustering installation directory. For example, using the recommended installation location, backups are stored in `/opt/continuent/backups`.

218
7.7.3. Creating an External Backup

There are several considerations to take into account when you are using a tool other than Tungsten Clustering 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 master then you may need to stop all access to the database.

- Are you taking a backup of the master?

  The Tungsten Replicator stores information in a schema to indicate the restart position for replication. On the master there can be a slight lag between this position and the actual position of the master. 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 master, you must track the actual binary log position of the master 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 master 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 master or a slave 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 Slave”.

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 Clustering 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-0000088804.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 offline
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 Clustering, 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 master server. There may be a difference between the binary log position of the master 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 master and find the correct sequence number for that position:

```
shell> ssh host1
shell> thl list -service alpha -low 32033674 -headers | grep 473863524
32033678 32030709 0 true 2014-10-17 16:58:11.0 mysql-bin.000032:0000000473863524;-1 db1-east.continuent.com
shell> exit
```

   - Return to the slave node and run `tungsten_set_position` to update the `trep_commit_seqno` table:

```
shell> tungsten_set_position --service=alpha --source=host1 --seqno=32033678
```

5. Put the replication service online using `trepctl`:

```
shell> trepctl online
```

7.8.3. Restoring from Another Slave

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 master or a slave 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 Slave".

Data can be restored to a slave by performing a backup on a different slave, transferring the backup information to the slave 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`:
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-0000000006-mysqldump-8120968634569665.sql host3:$PWD/
```

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 Slave

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 slave.

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/mysqld-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 slave may not work. To recover from another running slave, 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/mysqld stop
```

4. Stop the mysqld service on host3:

```
shell> sudo /etc/init.d/mysqld stop
```

5. Delete the mysqld data directory on host2:

```
```
6. If necessary, ensure the `tungsten` user can write to the MySQL directory:

   ```shell>
   sudo chown 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`:

    ```shell>
    sudo chown -R mysql:mysql /var/lib/mysql
    ```

    ```shell>
    sudo chmod 770 /var/lib/mysql
    ```

11. Clear out the THL files on the target node `host2` so the slave replicator service may start cleanly:

    ```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
    ```

7.9. Deploying Automatic Replicator Recovery

Automatic recovery enables the replicator to go back `ONLINE` [212] in the event of a transient failure that is triggered during either the `ONLINE` [212] or `GOING-ONLINE` [212] state that would otherwise trigger a change of states to `OFFLINE` [212]. For example, connection failures, or restarts in the MySQL service, trigger the replicator to go `OFFLINE` [212]. With autorecovery enabled, the replicator will attempt to put the replicator `ONLINE` [212] 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` [212] within the configurable reset interval, then the replicator will go to the `OFFLINE` [212] state.

3. If the replicator remains in the `ONLINE` [212] 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` [365]
  
  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` [365]
The delay between entering the `OFFLINE` 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` [365]

The duration after a successful autorecovery has been completed that the replicator must remain in the `ONLINE` 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` [365] 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. 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

#### 7.10.1.1. 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 Master/Slave 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 5, 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 environment.

• 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
top 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 Master/Slave 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
```
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-5.4.1-41
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

```
shell> dba法定代表
```

- Install the software

```
shell> cd /opt/continuent/software/tungsten-replicator-5.4.1-41
shell> tools/tpm install
```

- Finally start the replicator 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
```

7.10.1.2. Seeding Data from a Cluster, for a Cluster-Slave 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 Master/Slave 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-slave 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 5, 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-slave 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 master, or we can take an export from a slave.

  **Export from a Master:**

  - If you are able to export from the master, 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 Slave:**

  - To export from a slave, we cannot obtain the binlog position correctly as the one we need is specific to the Master, however, exporting from a slave means that we can utilise the features of Tungsten Clustering 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 slavehost 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 set -x
      dsctl set -segno 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 slavehost 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 environment.

- 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:

```bash
shell> replicator stop
```

We have now finished with the temporary MySQL instance.

**Step 7: Reconfigure the Applier as a Cluster-Slave**

• We now need to reconfigure the applier as a cluster slave, 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.

```bash
shell> cd /opt/continuent/software/tungsten-replicator-5.4.1-41
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

```bash
shell> cd /opt/continuent/software/tungsten-replicator-5.4.1-41
shell> tools/tpm install
```

• Finally start the replicator in an offline state:

```bash
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 slave and already have a `dsctl set` command, then move onto the next step. If you took an export from the MASTER then we need to retrieve the correct positions from the dump file.

Locate the dump file and issue the following command:

```bash
shell> grep "CHANGE MASTER" dump.sql
```

Taking the sequence value from `MASTER_LOG_FILE` filename and the `MASTER_LOG_POS`, issue the following statement on the MASTER host in the cluster, and copy the resulting dsctl command:

```bash
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 slave 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

```bash
shell> replicator start offline
```

### 7.11. Using the Parallel Extractor

Parallel Extractor in 2.2.1. The parallel extractor functionality was added in Tungsten Replicator 2.2.1, and supports only extraction from Oracle masters.

**Supported only for Oracle Extraction.** The `-provision` option is only supported for extraction from Oracle.

The parallel extractor reads information from the source database schema in chunks and then feeds this information into the THL data stream as row-based `INSERT` operations. When the slave connects, these are applied to the slave database as with a normal `INSERT` operations. The parallel extractor is particularly useful in heterogeneous environments such as Oracle to MySQL where the slave data does already exist on the slave.

The basic provisioning process operates in two stages:

1. Provisioning data is extracted and inserted into the THL. One event is used to contain all of the data from a single table. If the table is too large to be contained in a single event, the data will be distributed over multiple events.

2. Once provisioning has finished, data is extracted from the CDC as normal and added to the THL using the normal THL extraction thread.

This allows existing data to be extracted and processed through the replicator path, including filters within the applier. Once the initial data has been extracted, the change data to be applied. A diagram of the replication scheme at different stages is provided below:
The parallel extractor happens in a multi-threaded process that extracts multiple tables, and multiple ranges from a single table in parallel. A chunking thread identifies all the tables, and also identifies the keys and chunks that can be extracted from each table. It then coordinates the multiple threads:

- Multiple chunks from the source tables are extracted in parallel
- Multiple tables are extracted in parallel

For example, when reading from two different tables in a single schema, the process might look like the figure below:

Because multiple threads are used to read information from the tables, the process is very quick, although it implies additional load on the source database server, since the queries must load all of the data into memory.

To use the parallel extractor to provision data into the slave, the configuration must be performed as part of the installation process when configuring the master replicator for the first time, or when re-initializing the replicator on a master after a `trepctl reset` operation.

To setup provisioning with parallel extractor:

1. Install master Tungsten Replicator using `tpm`, but do not enable automatic starting (i.e. do not use the `--start` or `--start-and-repport` options).
2. Install the slave replicator as normal.
3. On the master:
a. Start the replicator in \texttt{OFFLINE} mode using \texttt{replicator start offline}:

\begin{verbatim}
shell> replicator start offline
\end{verbatim}

b. Put the replicator into the \texttt{ONLINE} state, using the \texttt{-provision} option:

\begin{verbatim}
shell> trepctl online -provision
\end{verbatim}

If you have an identifiable reference number, such as a the system change number or MySQL event, then this can be specified on the command-line to the \texttt{trepctl online -provision} command:

\begin{verbatim}
shell> trepctl online -provision 48748375
\end{verbatim}

During the provisioning process, the replicator will show the status \texttt{GOING-ONLINE:PROVISIONING} until all of the data has been read from the existing database.

The master will now start to read the information currently stored and feed this information through a separate pipeline into the THL.

4. On the slave, start the replicator, or put the replicator online. Statements from the master containing the provisioning information should be replicated into the slave.

\textbf{Important}

If the replicator is placed offline while the parallel extractor is still extracting data, the extraction process will continue to run and insert data until the extraction process has been completed.

Once the provisioned data has been inserted, replication will continue from the position where changes started to occur after the replicator was installed.

\textbf{7.11.1. Advanced Configuration Parameters}

The provisioning process can be controlled using a number of properties that can be configured when the replicator is installed by using \texttt{--property} option. For example:

\begin{verbatim}
shell> tpm update alpha \\
  --property=replicator.extractor.parallel-extractor.ChunkDefinitionFile=/opt/continuent/share/chunks.csv
\end{verbatim}

- \texttt{replicator.extractor.parallel-extractor.ChunkDefinitionFile}

The path to a file that contains a specification of the tables that should be included in the extraction. If no file is specified then all tables and extracted for provisioning.

The format of the chunk definition file is a Comma Separated Values (CSV) file, with each line containing the schema name, optional table name, optional chunk size and optional column list to be extracted. For example:

\begin{verbatim}
SALES
\end{verbatim}

Would extract all tables within the schema \texttt{SALES}:

\begin{verbatim}
SALES.INVOICES
\end{verbatim}

Would extract only the \texttt{INVOICES} table from the \texttt{SALES} schema.

\begin{verbatim}
SALES.INVOICES,1000
\end{verbatim}

Would extract only the \texttt{INVOICES} table from the \texttt{SALES} schema, only in chunks of 1000 rows.

To extract only specific columns, add the column list to the end of the schema, table and chunk size. For example:

\begin{verbatim}
SALES.INVOICES,1000,INVOICENO,CUSTOMERID,VALUE
\end{verbatim}

Multiple lines can be used to define a range of schemas, tables, and chunk size combinations.

- \texttt{replicator.extractor.parallel-extractor.chunk_size}

The chunk size defines the number of rows that are extracted from the source tables and inserted into the THL. This number should be adjusted when extracting very large rows from the source tables.

- \texttt{replicator.extractor.parallel-extractor.add_truncate_table}

If set to \texttt{true}, a \texttt{TRUNCATE} statement is inserted into the THL before the first row data is inserted into the THL. This empties the target table before the provision data is replicated.
• `replicator.extractor.parallel-extractor.extract_channels`

  Specifies the number of simultaneous threads that will be used to extract information. Defaults to a single thread.

• `replicator.extractor.parallel-extractor.queue_size`

  The number of events buffered inside the parallel extractor before they are inserted into the THL. Defaults to 20 transactions.

### 7.12. Switching Master Hosts

In the event of a failure, or during the process of performing maintenance on a running cluster, the roles of the master and slaves within the cluster may need to be swapped.

The basic sequence of operation for switching master and slaves is:

1. Switch slaves to offline state
2. Switch master to offline status
3. Set an existing slave to have the master role
4. Set each slave with the slave role, updating the master URI (where the THL logs will be loaded) to the new master host
5. Switch the new master to online state
6. Switch the new slaves 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 slave that will be promoted to the master role.

To perform an ordered switch of the master. In the example below, master host `host1` will be switched to `host3`, and the remaining hosts (`host1` and `host2`) will be configured as slaves to the new master:

1. If you are performing the switch as part of maintenance or other procedures, you should perform a safe switch, ensuring the slaves are up to date with the master:
   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 slaves 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 slave within the cluster, wait until the master sequence number has been reached, and then put the slave into the offline state:

      ```shell
      trepctl -host host2 wait -applied 1405
      trepctl -host host2 offline
      trepctl -host host3 wait -applied 1405
      trepctl -host host3 offline
      ```

      If the master has failed, or once the slaves and masters are in sync, you can perform the remainder of the steps to execute the physical switch.

2. Switch the master to the offline state:

   ```shell
   trepctl -host host1 offline
   ```

3. Configure the new designated master to the master role:

   ```shell
   trepctl -host host3 setrole -role master
   ```

   Switch the master to the online state:

   ```shell
   trepctl -host host3 online
   ```

4. For each slave, set the role to slave, supplying the URI of the THL service on the master:

   ```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 slave into the online state:

```shell
$ trepctl -host host1 online
```

Repeat for the remaining slaves:

```shell
$ trepctl -host host2 setrole -role slave -uri thl://host3:2112
$ 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
role               : slave
state              : ONLINE

appliedLastEventId : mysql-bin.000005:0000000000002100;0
appliedLastSeqno   : 1405
appliedLatency     : 0.089
dataServerHost     : host2
masterConnectUri   : thl://host3:2112
role               : slave
state              : ONLINE

appliedLastEventId : mysql-bin.000005:0000000000002100;0
appliedLastSeqno   : 1405
appliedLatency     : 0.149
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 slave to the `host1` as master.

### 7.13. 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 slaves lagging behind the master which can affect application performance. Parallel replication operates by distributing the events from the replication stream from different database schemas in parallel on the slave. 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:
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
dsyncOnFlush   : false
dlogConnectionTimeout : 28888
logDir         : /opt/continuent/thl/firstrep
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         : THLParallelWriteTask task_id=0 thread_name=store-thl-0
                hi_seqno=0 lo_seqno=0 read=0 accepted=0 discarded=0 events=0
store.1         : THLParallelWriteTask task_id=1 thread_name=store-thl-1
                hi_seqno=0 lo_seqno=0 read=0 accepted=0 discarded=0 events=0
store.2         : THLParallelWriteTask task_id=2 thread_name=store-thl-2
                hi_seqno=0 lo_seqno=0 read=0 accepted=0 discarded=0 events=0
store.3         : THLParallelWriteTask task_id=3 thread_name=store-thl-3
                hi_seqno=0 lo_seqno=0 read=0 accepted=0 discarded=0 events=0
store.4         : THLParallelWriteTask 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:

```
shell> trepctl status -name shards
Processing status command (shards)...
NAME           VALUE
---------       ---------
```

```
NAME           VALUE
---------       ---------
```

7.14. Performing Database or OS Maintenance

When performing database or operating system maintenance, datasources should be temporarily disabled by placing them into the `OF-LINE` state. For maintenance operations on a master, the current master should be switched, the required maintenance steps performed, and then the master switched back. Detailed steps are provided below for different scenarios.

To perform maintenance on a single slave, you should ensure that your application is not using the slave, perform the necessary maintenance, and then re-enable the slave 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.

7.14.2. Performing Maintenance on a Master

Maintenance, including MySQL admin or schema updates, should not be performed directly on a master as this may upset the replication and therefore availability and functionality of the slaves which are reading from the master.

To effectively make the modifications, you should switch the master host, then operate on the master as if it were slave, 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 master.

The complete sequence and commands required to perform maintenance on an active master are shown in the table below. The table assumes a dataservice with three datasources:

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Command</th>
<th>host1</th>
<th>host2</th>
<th>host3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Initial state</td>
<td></td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>2</td>
<td>Switch master to host2</td>
<td>See Section 7.12, “Switching Master Hosts”</td>
<td>Slave</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>3</td>
<td>Put slave into OFFLINE state</td>
<td>trepctl -host host1 offline</td>
<td>Offline</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>4</td>
<td>Perform maintenance</td>
<td></td>
<td>Offline</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>5</td>
<td>Validate the host1 server configuration</td>
<td>tpm validate</td>
<td>Offline</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>6</td>
<td>Put the slave online</td>
<td>trepctl -host host1 online</td>
<td>Slave</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>7</td>
<td>Ensure the slave has caught up</td>
<td>trepctl -host host1 status</td>
<td>Slave</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>8</td>
<td>Switch master back to host1</td>
<td>See Section 7.12, “Switching Master Hosts”</td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
</tbody>
</table>

7.14.3. 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 slaves
2. Switch the master to one of the already maintained slaves

3. Perform maintenance on the old master [now in slave state]

4. Switch the old master back to be the master 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 master, two slaves], but the same principles can be applied to any master/slave dataservice:

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Command</th>
<th>host1</th>
<th>host2</th>
<th>host3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Initial state</td>
<td></td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>2</td>
<td>Set the slave host2 offline</td>
<td>trepctl -host host2 offline</td>
<td>Master</td>
<td>Offline</td>
<td>Slave</td>
</tr>
<tr>
<td>3</td>
<td>Perform maintenance</td>
<td></td>
<td>Master</td>
<td>Offline</td>
<td>Slave</td>
</tr>
<tr>
<td>4</td>
<td>Validate the host2 server configuration</td>
<td>tpm validate</td>
<td>Master</td>
<td>Offline</td>
<td>Slave</td>
</tr>
<tr>
<td>5</td>
<td>Set slave host2 Online</td>
<td>trepctl -host host2 online</td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>6</td>
<td>Ensure the slave [host2] has caught up</td>
<td>trepctl -host host2 status</td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>7</td>
<td>Set the slave host3 offline</td>
<td>trepctl -host host3 offline</td>
<td>Master</td>
<td>Slave</td>
<td>Offline</td>
</tr>
<tr>
<td>8</td>
<td>Perform maintenance</td>
<td></td>
<td>Master</td>
<td>Slave</td>
<td>Offline</td>
</tr>
<tr>
<td>9</td>
<td>Validate the host3 server configuration</td>
<td>tpm validate</td>
<td>Master</td>
<td>Slave</td>
<td>Offline</td>
</tr>
<tr>
<td>10</td>
<td>Set the slave host3 online</td>
<td>trepctl -host host3 online</td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>11</td>
<td>Ensure the slave [host3] has caught up</td>
<td>trepctl -host host3 status</td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>12</td>
<td>Switch master to host2</td>
<td>See Section 7.12, “Switching Master Hosts”</td>
<td>Slave</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>13</td>
<td>Set the slave host1</td>
<td>trepctl -host host1 offline</td>
<td>Offline</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>14</td>
<td>Perform maintenance</td>
<td></td>
<td>Offline</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>15</td>
<td>Validate the host1 server configuration</td>
<td>tpm validate</td>
<td>Offline</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>16</td>
<td>Set the slave host1 online</td>
<td>trepctl -host host1 online</td>
<td>Slave</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>17</td>
<td>Ensure the slave [host1] has caught up</td>
<td>trepctl -host host1 status</td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>18</td>
<td>Switch master back to host1</td>
<td>See Section 7.12, “Switching Master Hosts”</td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
</tbody>
</table>

### 7.14.4. 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 Clustering 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 master switch and controlled stopping of services during the update process.

A sample sequence for this in a 3-node cluster is described below:

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Command</th>
<th>host1</th>
<th>host2</th>
<th>host3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Initial state</td>
<td></td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>2</td>
<td>Stop all services on host2.</td>
<td>stopall</td>
<td>Master</td>
<td>Stopped</td>
<td>Slave</td>
</tr>
<tr>
<td>3</td>
<td>Update the JVM</td>
<td></td>
<td>Master</td>
<td>Stopped</td>
<td>Slave</td>
</tr>
<tr>
<td>4</td>
<td>Start all services on host2 slave.</td>
<td>startall</td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>5</td>
<td>Stop all services on host3.</td>
<td>stopall</td>
<td>Master</td>
<td>Slave</td>
<td>Stopped</td>
</tr>
<tr>
<td>Step</td>
<td>Description</td>
<td>Command</td>
<td>host1</td>
<td>host2</td>
<td>host3</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>---------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>6</td>
<td>Update the JVM</td>
<td>Master</td>
<td>Slave</td>
<td>Stopped</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Start all services on host3 slave.</td>
<td>startall</td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>8</td>
<td>Stop all services on host1.</td>
<td>stopall</td>
<td>Stopped</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>9</td>
<td>Update the JVM</td>
<td>Stopped</td>
<td>Slave</td>
<td>Slave</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Start all services on host1 Master.</td>
<td>startall</td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
</tbody>
</table>

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.

### 7.15. 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.

For installations using Tungsten Replicator 2.1.1 and later where `tpm` has been used to perform the installation, use the instructions in Section 7.15.1, "Upgrading Tungsten Replicator using tpm."

#### 7.15.1. 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-5.4.1-41.tar.gz
```

3. Change to the unpackaged directory:
   ```shell
cd tungsten-replicator-5.4.1-41
```

4. Run the validation process:
   ```shell
./tools/tpm validate-update
```

   **Note**
   The validate process will check that prerequisities are in place and that `tpm` can safely upgrade the software. Any errors that are reported should be handled before proceeding.

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:
2. Unpack the release package:
```
shell> tar zxf tungsten-replicator-5.4.1-41.tar.gz
```

3. Change to the unpackaged directory:
```
shell> cd tungsten-replicator-5.4.1-41
```

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` [382]
  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` [403]
  The username to be used when logging in to other hosts.

- `--directory` [378]
  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-requisites are in place and that `tpm` can safely upgrade the software on all hosts. Any errors that are reported should be handled before proceeding.

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`.

### 7.15.2. 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.
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
```

7.15.3. 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.

7.16. Monitoring Tungsten Clustering

It is your responsibility to properly monitor your deployments of Tungsten Clustering 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 Clustering and Tungsten Replicator services are running.
2. Make sure all datasources and replication services are **ONLINE**.
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 Clustering 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.

### 7.16.1. Managing Log Files with logrotate

You can manage the logs generated by Tungsten Clustering using logrotate.

- trepsvc.log

```
/opt/continuent/tungsten/tungsten-replicator/log/trepsvc.log {
  notifempty
  daily
  rotate 3
  missingok
  compress
  copytruncate
}
```

### 7.16.2. 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](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 Clustering. If you are using multiple replication 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.
7.16.3. Monitoring Status Using nagios

In addition to the scripts bundled with the software, there is a Ruby gem available with expanded checks and a mechanism to add custom checks. See https://github.com/continuent/continuent-monitors-nagios for more details.
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
      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.

### 7.17. Rebuilding THL on the Master

If THL is lost on a master before the events contained within it have been applied to the slave[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 master and slaves is:

1. Gather the failing requested sequence numbers from all slaves:
In the above example, when slave db2 comes back online, it requests a copy of the last seqno in local thl (4) from the master db1 to compare for data integrity purposes, which the master no longer has.

Keep a note of the lowest sequence number and the host that it is on across all slaves for use in the next step.

2. On the slave 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 [271] specifying the sequence number above. This information will be needed on the extractor [master] in a later step. For example:

```
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, 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> tungsten_get_position
{
  "applied_latency": 0,
  "epoch_number": 0,
  "eventid": "mysql-bin.000009:0000000000001844;56",
  "extract_timestamp": "2017-07-14 14:49:00.0",
  "fragno": 0,
  "last_frag": true,
  "seqno": 4,
  "shard_id": "#UNKNOWN",
  "source_id": "db1",
  "task_id": 0,
  "update_timestamp": "2017-07-14 14:49:00.0"
}
```

3. Clear all THL on the master since it is no longer needed by any slaves:

```
shell> thl purge
```

4. Use the tungsten_set_position command on the master with the values we got from the slave with the lowest seqno to tell the master replicator to begin generating THL starting from that event in the MySQL binary logs:

```
shell> tungsten_set_position --seqno=4 --epoch=0 --source-id=db1 --event-id=mysql-bin.00009:0000000000001844
```

You may also use dsctl, but that requires executing the dsctl reset command first.

5. Switch the master to online state:

```
shell> trepctl online
```

6. Switch the slaves to online state once the master is fully online:

```
shell> trepctl online
```
Chapter 8. Command-line Tools

Tungsten Clustering 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 Clustering 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
- **setupCDC.sh** — Setup Oracle Change Data Control services
- **updateCDC.sh** — Update an existing Oracle Change Data Control service

Commands related to the core Tungsten Replicator
- **trepc1** — replicator control
- **multi_trepc1** — multi-replicator control
- **thl** — examine Tungsten History Log contents

Commands related to managing Tungsten Replicator deployments
- **tungsten_provision_slave** — provision or reprovision a slave from an existing master or slave database
- **tungsten_read_master_events** — read master events to determine the correct log position
- **tungsten_set_position** — set the position of the replicator

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

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:

```
shells ls -l /opt/continuent/releases
drwxr-xr-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 15:36 tungsten-replicator-5.2.0-218_pid16197/
drwxr-xr-x  5 mc mc 4096 Jul 7 15:36 tungsten-replicator-5.2.0-219_pid16301/
drwxr-xr-x  5 mc mc 4096 Jul 7 15:36 tungsten-replicator-5.2.0-219_pid1393/
drwxr-xr-x  5 mc mc 4096 Jul 7 15:36 tungsten-replicator-5.2.0-219_pid24935/
drwxr-xr-x  5 mc mc 4096 Jul 7 15:36 tungsten-replicator-5.2.0-219_pid26720/
drwxr-xr-x  5 mc mc 4096 Jul 7 15:36 tungsten-replicator-5.2.0-219_pid28491/
drwxr-xr-x  5 mc mc 4096 Jul 7 15:36 tungsten-replicator-5.2.0-219_pid30270/
drwxr-xr-x  5 mc mc 4096 Jul 7 15:36 tungsten-replicator-5.2.0-219_pid32041/
drwxr-xr-x  5 mc mc 4096 Jul 7 15:36 tungsten-replicator-5.2.0-219_pid4983/
drwxr-xr-x  5 mc mc 4096 Jul 7 15:36 tungsten-replicator-5.2.0-219_pid6754/
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 Clustering.
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_pid30278
Deleting /opt/continuent/releases/tungsten-replicator-5.2.0-219_pid249491
Deleting /opt/continuent/releases/tungsten-replicator-5.2.0-219_pid16728
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-5.2.0-218_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_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>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-c</code></td>
<td>Report a critical status if the latency is above this level</td>
</tr>
<tr>
<td><code>--perfdata</code></td>
<td>Show the latency performance information</td>
</tr>
<tr>
<td><code>--perslave-perfdata</code></td>
<td>Show the latency performance information on a per-slave basis</td>
</tr>
<tr>
<td><code>-w</code></td>
<td>Report a warning status if the latency is above this level</td>
</tr>
</tbody>
</table>

Table 8.1. `check_tungsten_latency` Options

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:
8.3. The check_tungsten_online Command

The **check_tungsten_online** command checks whether all the hosts in a given service are online and running.

**Table 8.2. check_tungsten_online Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-h</code></td>
<td>Display the help text</td>
</tr>
<tr>
<td><code>-port</code></td>
<td>RMI port for the replicator being checked</td>
</tr>
</tbody>
</table>

This command only needs to be run on one node within the service; the command returns the status for all nodes.

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:
  
  ```shell>
  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:
  
  ```shell>
  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**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-h</code></td>
<td>Display the help text</td>
</tr>
<tr>
<td><code>-r</code></td>
<td>Check the replication services status.</td>
</tr>
</tbody>
</table>

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:

```
shell> sudo deployall
```

The startup scripts are added to the correct run levels to enable operation during standard startup and shutdown levels.

See Section 2.6, “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 2.9, “Understanding Heterogeneous Deployments”.

For example, to generate Oracle DDL from an existing MySQL database:

```
shell> ddlscan -user tungsten -url jdbc:mysql:thin://host1:13306/test -pass password -template ddl-mysql-oracle.vm -db test
```

SQL generated on Thu Sep 11 15:39:06 BST 2014 by ./ddlscan utility of Tungsten

```
url = jdbc:mysql:thin://host1:13306/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:

```
```
Command-line Tools

The available options are as follows:

**Table 8.4. ddlscan Command-line Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-conf path [246]</td>
<td>Path to a static-{svc}.properties file to read JDBC connection address and credentials</td>
</tr>
<tr>
<td>-db db [246]</td>
<td>Database to use [will substitute ${DBNAME} in the URL, if needed]</td>
</tr>
<tr>
<td>-opt opt val [247]</td>
<td>Option(s) to pass to template, try: -opt help me</td>
</tr>
<tr>
<td>-out file [247]</td>
<td>Render to file [print to stdout if not specified]</td>
</tr>
<tr>
<td>-pass secret [246]</td>
<td>JDBC password</td>
</tr>
<tr>
<td>-path path [247]</td>
<td>Add additional search path for loading Velocity templates</td>
</tr>
<tr>
<td>-rename file [247]</td>
<td>Definitions file for renaming schemas, tables and columns</td>
</tr>
<tr>
<td>-service name [246]</td>
<td>Name of a replication service instead of path to config</td>
</tr>
<tr>
<td>-tableFile file [246]</td>
<td>New-line separated definitions file of tables to find</td>
</tr>
<tr>
<td>-tables regex [246]</td>
<td>Comma-separated list of tables to find</td>
</tr>
<tr>
<td>-template file [246]</td>
<td>Specify template file to render</td>
</tr>
<tr>
<td>-url jdbcUrl [246]</td>
<td>JDBC connection string [use single quotes to escape]</td>
</tr>
<tr>
<td>-user user [246]</td>
<td>JDBC username</td>
</tr>
</tbody>
</table>

DDL scan 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 [246] 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 [246] 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
```

dlscan 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 [246]**

  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 [246]**
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> dddlscan -service alpha -template dddl-mysql-oracle.vm -db test -tableFile tablelist.txt
```

- `rename` [247]

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` [247]

The path to additional Velocity templates to be searched when specifying the template name.

- `opt` [247]

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> dddlscan -service alpha -template dddl-mysql-oracle.vm -db test -opt schemaPrefix mysql_
```

- `out` [247]

Sends the generated DDL output to a file, in place of sending it to standard output.

- `help` [247]

Generates the help text of arguments.

### 8.6.2. Supported Templates and Usage

**Table 8.5. dddlscan Supported Templates**

<table>
<thead>
<tr>
<th>file</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dddl-check-pkeys.vm</td>
<td>Reports which tables are without primary key definitions</td>
</tr>
<tr>
<td>dddl-mysql-cassandra.vm</td>
<td>Generates DDL from a MySQL host suitable for the base tables in Apache Cassandra</td>
</tr>
<tr>
<td>dddl-mysql-cassandra-staging.vm</td>
<td>Generates DDL from a MySQL host suitable for the base tables in Apache Cassandra</td>
</tr>
<tr>
<td>dddl-mysql-hive-0.10.vm</td>
<td>Generates DDL from a MySQL host suitable for the base tables in a Hadoop/Hive Environment</td>
</tr>
<tr>
<td>dddl-mysql-hive-0.10-staging.vm</td>
<td>Generates DDL from a MySQL host suitable for the staging tables in a Hadoop/Hive Environment</td>
</tr>
<tr>
<td>dddl-mysql-hive-metadata.vm</td>
<td>Generates metadata as JSON to be used within a Hadoop/Hive Environment</td>
</tr>
<tr>
<td>dddl-mysql-oracle.vm</td>
<td>Generates Oracle schema from a MySQL schema</td>
</tr>
<tr>
<td>dddl-mysql-oracle-cdc.vm</td>
<td>Generates Oracle tables with CDC capture information from a MySQL schema</td>
</tr>
<tr>
<td>dddl-mysql-redshift.vm</td>
<td>Generates DDL from a MySQL host suitable for the base tables in Amazon Redshift.</td>
</tr>
<tr>
<td>dddl-mysql-redshift-staging.vm</td>
<td>Generates DDL from a MySQL host suitable for the staging tables in Amazon Redshift.</td>
</tr>
<tr>
<td>dddl-mysql-vertica.vm</td>
<td>Generates DDL suitable for the base tables in HP Vertica</td>
</tr>
<tr>
<td>dddl-mysql-vertica-staging.vm</td>
<td>Generates DDL suitable for the staging tables in HP Vertica</td>
</tr>
<tr>
<td>ddl-oracle-mysql.vm</td>
<td>Generates DDL for MySQL tables from an Oracle schema</td>
</tr>
<tr>
<td>ddl-oracle-mysql-pk-only.vm</td>
<td>Generates Primary Key DDL statements from an Oracle database for MySQL</td>
</tr>
</tbody>
</table>
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
```

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-cassandra.vm**

The *ddl-mysql-cassandra.vm* template generates DDL for Apache Cassandra tables from MySQL schemas. For example:

```sql
CREATE TABLE "test"."all_mysql_types"(
  "my_id" INT ,
  "my_bit" BOOLEAN /* BIT(1) */ ,
  "my_tinyint" INT ,
  "my_boolean" INT ,
  "my_smallint" INT ,
  "my_mediumint" INT ,
  "my_int" INT ,
  "my_bigint" BIGINT ,
  "my_decimal_10_5" DECIMAL ,
  "my_float" FLOAT ,
  "my_double" DOUBLE ,
  "my_date" DATE ,
  "my_datetime" TIMESTAMPTZ ,
  "my_timestamp" TIMESTAMP ,
  "my_time" TIME ,
  "my_year" DECIMAL ,
  "my_char_10" TEXT ,
  "my_varchar_10" TEXT ,
  "my_tinytext" TEXT ,
  "my_text" TEXT ,
  "my_mediumtext" TEXT ,
  "my_longtext" TEXT ,
  "my_enum_abc" TEXT ,
  "my_set_def" TEXT ,
  primary key ("my_id")
);
```

Columns are translated as follows:

<table>
<thead>
<tr>
<th>Oracle Datatype</th>
<th>Cassandra Datatype</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIGINT</td>
<td>BIGINT</td>
</tr>
<tr>
<td>BIT</td>
<td>BOOLEAN</td>
</tr>
<tr>
<td>BOOLEAN</td>
<td>INT</td>
</tr>
<tr>
<td>CHAR</td>
<td>TEXT</td>
</tr>
<tr>
<td>DATE</td>
<td>DATE</td>
</tr>
<tr>
<td>DATETIME</td>
<td>TIMESTAMP</td>
</tr>
</tbody>
</table>
The following additional considerations are taken into account:

- The primary key generated is based on the primary key for the table and the opcode column used during processing. This provides optimal performance during loading.
- All column names are quoted (using double quotes) to prevent reserved word issues.
- Cassandra has significantly fewer types and so data and time values are often translated to simpler types.

### 8.6.2.3. `ddl-mysql-cassandra-staging.vm`

Creates a table suitable for use within Cassandra as a staging table for data from MySQL.

```sql
CREATE TABLE "test"."stage_xxx_all_mysql_types"
{
    "tungsten_opcode" TEXT ,
    "tungsten_seqno" INT ,
    "tungsten_row_id" INT ,
    "tungsten_commit_timestamp" TIMESTAMP ,
    "my_id" INT ,
    "my_bit" BOOLEAN /* BIT(1) */ ,
    "my_tinyint" INT ,
    "my_boolean" INT ,
    "my_smallint" INT ,
    "my_mediumint" INT ,
    "my_int" INT ,
    "my_bigint" BIGINT ,
    "my_decimal_10_5" DECIMAL ,
    "my_float" FLOAT ,
    "my_double" DOUBLE ,
    "my_date" DATE ,
    "my_datetime" TIMESTAMP ,
    "my_timestamp" TIMESTAMP ,
    "my_time" TIME ,
    "my_year" DECIMAL ,
    "my_char_10" TEXT ,
    "my_varchar_10" TEXT ,
    "my_tinytext" TEXT ,
    "my_text" TEXT ,
    "my_mediumtext" TEXT ,
    "my_longtext" TEXT ,
    "my_enum_abc" TEXT ,
    "my_set_def" TEXT ,
    primary key (tungsten_opcode,"my_id" ))
};
```

All datatype translates are identical to those used in `ddl_mysql-cassandra.vm`.

---

**Oracle Datatype** | **Cassandra Datatype**
---|---
DECIMAL | DECIMAL
DOUBLE | DOUBLE
ENUM | TEXT
FLOAT | FLOAT
INT | INT
LONGTEXT | TEXT
MEDIUMINT | INT
MEDIUMTEXT | TEXT
SET | TEXT
SMALLINT | INT
TEXT | TEXT
TIME | TIME
TIMESTAMP | TIMESTAMP
TINYINT | INT
TINYTEXT | TEXT
VARCHAR | TEXT
YEAR | DECIMAL
8.6.2.4. `ddl-mysql-hive-0.10.vm`

Generates DDL suitable for a carbon-copy form of the table from the MySQL host:

```shell
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:

<table>
<thead>
<tr>
<th>MySQL Datatype</th>
<th>Hive Datatype</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATETIME</td>
<td>STRING</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>TIMESTAMP</td>
</tr>
<tr>
<td>DATE</td>
<td>STRING</td>
</tr>
<tr>
<td>YEAR</td>
<td>INT</td>
</tr>
<tr>
<td>TIME</td>
<td>STRING</td>
</tr>
<tr>
<td>TINYINT</td>
<td>TINYINT</td>
</tr>
<tr>
<td>TINYINT UNSIGNED</td>
<td>SMALLINT</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>SMALLINT</td>
</tr>
<tr>
<td>SMALLINT UNSIGNED</td>
<td>SMALLINT</td>
</tr>
<tr>
<td>MEDIUMINT</td>
<td>INT</td>
</tr>
<tr>
<td>INT</td>
<td>INT</td>
</tr>
<tr>
<td>INT UNSIGNED</td>
<td>BIGINT</td>
</tr>
<tr>
<td>BIGINT</td>
<td>BIGINT</td>
</tr>
<tr>
<td>BIGINT UNSIGNED</td>
<td>STRING</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>STRING</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>STRING</td>
</tr>
<tr>
<td>CHAR</td>
<td>STRING</td>
</tr>
<tr>
<td>BINARY</td>
<td>BINARY</td>
</tr>
<tr>
<td>VARCHARINARY</td>
<td>BINARY</td>
</tr>
<tr>
<td>TEXT</td>
<td>STRING</td>
</tr>
<tr>
<td>BLOB</td>
<td>BINARY</td>
</tr>
<tr>
<td>FLOAT</td>
<td>DOUBLE</td>
</tr>
<tr>
<td>DOUBLE</td>
<td>DOUBLE</td>
</tr>
<tr>
<td>ENUM</td>
<td>STRING</td>
</tr>
<tr>
<td>SET</td>
<td>STRING</td>
</tr>
<tr>
<td>BIT</td>
<td>STRING</td>
</tr>
</tbody>
</table>

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_`.
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`.

### 8.6.2.5. 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 | 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:

```
 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.

### 8.6.2.6. 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:

```
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' ;
```
### 8.6.2.7. `ddl-mysql-oracle.vm`

When translating MySQL tables to Oracle compatible schema, the following datatypes are migrated to their closest Oracle equivalent:

<table>
<thead>
<tr>
<th>MySQL Datatype</th>
<th>Oracle Datatype</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT</td>
<td>NUMBER(10, 0)</td>
</tr>
<tr>
<td>BIGINT</td>
<td>NUMBER(19, 0)</td>
</tr>
<tr>
<td>TINYINT</td>
<td>NUMBER(3, 0)</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>NUMBER(5, 0)</td>
</tr>
<tr>
<td>MEDIUMINT</td>
<td>NUMBER(7, 0)</td>
</tr>
<tr>
<td>DECIMAL(x,y)</td>
<td>NUMBER(x, y)</td>
</tr>
<tr>
<td>FLOAT</td>
<td>FLOAT</td>
</tr>
<tr>
<td>CHAR(n)</td>
<td>CHAR(n)</td>
</tr>
<tr>
<td>VARCHAR(n)</td>
<td>VARCHAR2(n) ( n &lt; 2000 ), CLOB ( n &gt; 2000 )</td>
</tr>
<tr>
<td>DATE</td>
<td>DATE</td>
</tr>
<tr>
<td>DATETIME</td>
<td>DATE</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>DATE</td>
</tr>
<tr>
<td>TEXT</td>
<td>CLOB</td>
</tr>
<tr>
<td>BLOB</td>
<td>BLOB</td>
</tr>
<tr>
<td>ENUM(...)</td>
<td>VARCHAR(255)</td>
</tr>
<tr>
<td>ENUM(...)</td>
<td>VARCHAR(4000)</td>
</tr>
<tr>
<td>BIT(1)</td>
<td>NUMBER(1)</td>
</tr>
</tbody>
</table>

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.

### 8.6.2.8. `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 d6l-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`.

8.6.2.9. `ddl-mysql-redshift.vm`

The `ddl-mysql-redshift.vm` template generates DDL for Amazon Redshift tables from MySQL schemas. For example:

```sql
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) */
};
```

Columns are translated as follows:

<table>
<thead>
<tr>
<th>Oracle Datatype</th>
<th>Redshift Datatype</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIGINT</td>
<td>BIGINT</td>
</tr>
<tr>
<td>BINARY</td>
<td>BINARY, CHAR in 5.2.1 and later</td>
</tr>
<tr>
<td>BIT(1)</td>
<td>BOOLEAN</td>
</tr>
<tr>
<td>BIT</td>
<td>CHAR</td>
</tr>
<tr>
<td>BLOB</td>
<td>VARBINARY VARCHAR in 5.2.1 and later</td>
</tr>
<tr>
<td>CHAR</td>
<td>DATE</td>
</tr>
<tr>
<td>DATE</td>
<td>DATE</td>
</tr>
<tr>
<td>DATETIME</td>
<td>DATETIME</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>DECIMAL</td>
</tr>
<tr>
<td>DOUBLE</td>
<td>DOUBLE PRECISION</td>
</tr>
<tr>
<td>ENUM</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>FLOAT</td>
<td>FLOAT</td>
</tr>
<tr>
<td>INT</td>
<td>INT</td>
</tr>
</tbody>
</table>
## Oracle Datatype | Redshift Datatype
---|---
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 | VARCHAR
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.

### 8.6.2.10. `ddl-mysql-redshift-staging.vm`

The `ddl-mysql-redshift-staging.vm` template generates DDL for Amazon Redshift tables from MySQL schemas. For example:

```sql
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`.

### 8.6.2.11. `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
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

```sql
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:

<table>
<thead>
<tr>
<th>MySQL Datatype</th>
<th>Vertica Datatype</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATETIME</td>
<td>DATETIME</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>TIMESTAMP</td>
</tr>
<tr>
<td>DATE</td>
<td>DATE</td>
</tr>
<tr>
<td>TIME</td>
<td>TIME</td>
</tr>
<tr>
<td>TINYINT</td>
<td>TINYINT</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>SMALLINT</td>
</tr>
<tr>
<td>MEDIUMINT</td>
<td>INT</td>
</tr>
<tr>
<td>INT</td>
<td>INT</td>
</tr>
<tr>
<td>BIGINT</td>
<td>INT</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>CHAR</td>
<td>CHAR</td>
</tr>
<tr>
<td>BINARY</td>
<td>BINARY</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>TEXT, TINYTEXT, MEDIUMTEXT, LONGETEXT</td>
<td>VARCHAR(65000)</td>
</tr>
<tr>
<td>BLOB, TINYBLOB, MEDIUMBLOB, LONGLOB</td>
<td>VARBINARY(65000)</td>
</tr>
<tr>
<td>FLOAT</td>
<td>FLOAT</td>
</tr>
<tr>
<td>DOUBLE</td>
<td>DOUBLE PRECISION</td>
</tr>
<tr>
<td>ENUM</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>SET</td>
<td>VARCHAR(4000)</td>
</tr>
<tr>
<td>BIT(1)</td>
<td>BOOLEAN</td>
</tr>
<tr>
<td>BIT</td>
<td>CHAR(64)</td>
</tr>
</tbody>
</table>

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.
Command-line Tools

- **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 `enumbtostring` filter.

### 8.6.2.12.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
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;
```

### 8.6.2.13. 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
shell> ddlscan -service loans -template ddl-oracle-mysql.vm -db loans
/*
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;
```

### Columns are translated as follows:

<table>
<thead>
<tr>
<th>Oracle Datatype</th>
<th>MySQL Datatype</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE</td>
<td>DATETIME</td>
</tr>
<tr>
<td>NUMBER(8)</td>
<td>NUMERIC</td>
</tr>
<tr>
<td>NUMBER(n) where n &lt; 19</td>
<td>INT</td>
</tr>
<tr>
<td>NUMBER(n) where n &gt; 19</td>
<td>BIGINT</td>
</tr>
<tr>
<td>NUMBER(n) where n &lt; 3</td>
<td>TINYINT</td>
</tr>
<tr>
<td>NUMBER(n) where n &lt; 5</td>
<td>SMALLINT</td>
</tr>
</tbody>
</table>
### Command-line Tools

<table>
<thead>
<tr>
<th>Oracle Datatype</th>
<th>MySQL Datatype</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMBER(n) where n &lt; 7</td>
<td>MEDIUMINT</td>
</tr>
<tr>
<td>NUMBER(n) where n &lt; 10</td>
<td>INT</td>
</tr>
<tr>
<td>NUMBER(n) where n &lt; 19</td>
<td>BIGINT</td>
</tr>
<tr>
<td>NUMBER</td>
<td>DECIMAL</td>
</tr>
<tr>
<td>FLOAT</td>
<td>FLOAT</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>LONG</td>
<td>LONGTEXT</td>
</tr>
<tr>
<td>BFILE</td>
<td>VARCHAR(1024)</td>
</tr>
<tr>
<td>CHAR</td>
<td>CHAR</td>
</tr>
<tr>
<td>CLOB</td>
<td>LONGTEXT</td>
</tr>
<tr>
<td>BLOB</td>
<td>LONGBLOB</td>
</tr>
<tr>
<td>LONG RAW</td>
<td>LONGBLOB</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>TIMESTAMP</td>
</tr>
<tr>
<td>RAW</td>
<td>VARBINARY</td>
</tr>
</tbody>
</table>

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.

### 8.6.2.14. `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
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

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>get</td>
<td>Return the available position information</td>
</tr>
<tr>
<td>help</td>
<td>Print the help display</td>
</tr>
<tr>
<td>reset</td>
<td>Clear the datasource position information</td>
</tr>
<tr>
<td>set</td>
<td>Set the position</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-conf</code></td>
<td>Path to the static services properties file</td>
</tr>
<tr>
<td><code>-ds</code></td>
<td>Name of the datasource</td>
</tr>
<tr>
<td><code>-service</code></td>
<td>Name of the replication service to get information from</td>
</tr>
</tbody>
</table>

If more than one service or datasource has been configured, one of these options must 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

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-ascmd</code></td>
<td>Generates the command required to set the datasource to the current position</td>
</tr>
</tbody>
</table>

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",
  "event_id" : "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

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-epoch</code></td>
<td>Epoch Number</td>
</tr>
<tr>
<td><code>-event-id</code></td>
<td>Source Event ID</td>
</tr>
<tr>
<td><code>-reset</code></td>
<td>Resets the datasources before performing set operation</td>
</tr>
<tr>
<td><code>-seqno</code></td>
<td>Sequence number</td>
</tr>
<tr>
<td><code>-source-id</code></td>
<td>Source ID</td>
</tr>
</tbody>
</table>

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.00014:0000000000005645;-1" -source-id tr-11
```
Command-line Tools

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
```

Adding the `-reset` option to the `dsctl get -ascmd` command also adds the option to the generated command:

```
shell> dsctl get -ascmd -reset
```

8.7.3. dsctl reset Command

Clears the current replicator status and position information:

```
shell> dsctl reset
```

8.7.4. dsctl help Command

Displays the current help text:

```
shell> dsctl help
```

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
```

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-replicator/bin/trepctl'
```
8.9. The load-reduce-check Tool

The load-reduce-check tool is not part of the standard replicator distribution. The tool is part of the continuent-tools-hadoop repository, available from Github.

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.

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

The materialize tool is not part of the standard replicator distribution. The tool is part of the continuent-tools-hadoop repository, available from Github.

8.11. The multi_trepctl Command

The multi_trepctl command provides unified status and operation support across your Tungsten Clustering installation across multiple hosts without the need to run the trepctl command across multiple hosts and/or services individually.
multi_trepctl
backups [ --by-service ] [ --fields appliedLastSeqNo appliedLatency host role service Name state ]
heartbeat [ --host, --hosts self ]
list
masterof [ --output json list name tab yaml ] [ --path, --paths ] [ --role, --roles ]
runt [ --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
<table>
<thead>
<tr>
<th>host</th>
<th>serviceName</th>
<th>role</th>
<th>state</th>
<th>appliedLastSeqno</th>
<th>appliedLatency</th>
</tr>
</thead>
<tbody>
<tr>
<td>tr-ms1</td>
<td>alpha</td>
<td>master</td>
<td>ONLINE</td>
<td>54</td>
<td>0.867</td>
</tr>
<tr>
<td>tr-ms2</td>
<td>alpha</td>
<td>slave</td>
<td>ONLINE</td>
<td>54</td>
<td>1.746</td>
</tr>
<tr>
<td>tr-ms3</td>
<td>alpha</td>
<td>slave</td>
<td>ONLINE</td>
<td>54</td>
<td>42.851</td>
</tr>
</tbody>
</table>
```

On a server with multiple services, information is output for each service and host:

```
shell> multi_trepctl
<table>
<thead>
<tr>
<th>host</th>
<th>servicename</th>
<th>role</th>
<th>state</th>
<th>appliedLastSeqno</th>
<th>appliedLatency</th>
</tr>
</thead>
<tbody>
<tr>
<td>east1</td>
<td>east</td>
<td>master</td>
<td>ONLINE</td>
<td>64</td>
<td>59.380</td>
</tr>
<tr>
<td>east1</td>
<td>east</td>
<td>slave</td>
<td>ONLINE</td>
<td>64</td>
<td>60.899</td>
</tr>
<tr>
<td>west1</td>
<td>east</td>
<td>master</td>
<td>ONLINE</td>
<td>294328</td>
<td>0.319</td>
</tr>
<tr>
<td>west1</td>
<td>east</td>
<td>slave</td>
<td>ONLINE</td>
<td>331595</td>
<td>0.316</td>
</tr>
<tr>
<td>west2</td>
<td>east</td>
<td>slave</td>
<td>ONLINE</td>
<td>294328</td>
<td>119.834</td>
</tr>
<tr>
<td>west2</td>
<td>east</td>
<td>slave</td>
<td>ONLINE</td>
<td>231595</td>
<td>181.128</td>
</tr>
<tr>
<td>west3</td>
<td>east</td>
<td>slave</td>
<td>ONLINE</td>
<td>231595</td>
<td>204.790</td>
</tr>
<tr>
<td>west3</td>
<td>east</td>
<td>slave</td>
<td>ONLINE</td>
<td>231595</td>
<td>22.895</td>
</tr>
</tbody>
</table>
```

8.11.1. 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 8.10. multi_trepctl Command-line Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--by-service [261]</td>
<td>Sort the output by the service name</td>
</tr>
<tr>
<td>--fields [261]</td>
<td>Fields to be output during summary</td>
</tr>
<tr>
<td>--host [262], --hosts [262]</td>
<td>Host or hosts on which to limit output</td>
</tr>
<tr>
<td>--output [262]</td>
<td>Specify the output format</td>
</tr>
<tr>
<td>--paths [262], --path [262]</td>
<td>Directory or directories to check when looking for tools</td>
</tr>
<tr>
<td>--role [263], --roles [263]</td>
<td>Role or roles on which to limit output</td>
</tr>
<tr>
<td>--service [263], --services [263]</td>
<td>Service or services on which to limit output</td>
</tr>
<tr>
<td>--skip-headers [263]</td>
<td>Skip the headers</td>
</tr>
<tr>
<td>--sort-by [263]</td>
<td>Sort by a specified field</td>
</tr>
</tbody>
</table>

Where:

- • --by-service [261]
  Order the output according to the service name and role within the service:

```
shell> multi_trepctl --by-service
<table>
<thead>
<tr>
<th>host</th>
<th>servicename</th>
<th>role</th>
<th>state</th>
<th>appliedLastSeqno</th>
<th>appliedLatency</th>
</tr>
</thead>
<tbody>
<tr>
<td>east1</td>
<td>east</td>
<td>master</td>
<td>ONLINE</td>
<td>64</td>
<td>59.380</td>
</tr>
<tr>
<td>west1</td>
<td>east</td>
<td>master</td>
<td>ONLINE</td>
<td>64</td>
<td>60.899</td>
</tr>
<tr>
<td>west1</td>
<td>east</td>
<td>slave</td>
<td>ONLINE</td>
<td>64</td>
<td>68.979</td>
</tr>
<tr>
<td>west1</td>
<td>west</td>
<td>master</td>
<td>ONLINE</td>
<td>294328</td>
<td>8.319</td>
</tr>
<tr>
<td>west2</td>
<td>west</td>
<td>master</td>
<td>ONLINE</td>
<td>231595</td>
<td>8.316</td>
</tr>
<tr>
<td>east1</td>
<td>west</td>
<td>master</td>
<td>ONLINE</td>
<td>331595</td>
<td>119.834</td>
</tr>
<tr>
<td>west1</td>
<td>west</td>
<td>slave</td>
<td>ONLINE</td>
<td>294328</td>
<td>181.128</td>
</tr>
<tr>
<td>west2</td>
<td>west</td>
<td>slave</td>
<td>ONLINE</td>
<td>231595</td>
<td>204.790</td>
</tr>
</tbody>
</table>
```

- • --fields [261]
  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;

```
Command-line Tools

```
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 [262], --hosts [262]**

  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 [262]**

  Specify the output format.

  **Table 8.11. multi_trepctl --output Option**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
<th>Default</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>--output</td>
<td>Specify the output format</td>
<td>string</td>
<td>info</td>
<td>json, list, name, tab, yaml</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>JSON format, List format, Name [simplified text] format, Tab-delimited format, YAML format</td>
</tr>
</tbody>
</table>

  For example, to output the current status in JSON format:

  ```
  shell> multi_trepctl --output json
  [
    {
      "appliedlastseqno": 2322,
      "appliedlatency": 0.524,
      "host": "tr-ms1",
      "servicename": "alpha",
      "state": "ONLINE"
    },
    {
      "appliedlastseqno": 2322,
      "appliedlatency": 0.0,
      "host": "tr-ms2",
      "servicename": "alpha",
      "state": "ONLINE"
    },
    {
      "appliedlastseqno": -1,
      "appliedlatency": -1.0,
      "host": "tr-ms3",
      "servicename": "alpha",
      "state": "OFFLINE:ERROR"
    }
  ]
  ```

- **--path [262], --paths [262]**

  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 |          0.451 |
  | db5  | east        | slave | ONLINE |                4 |          0.460 |
  | db6  | west        | slave | ONLINE |                3 |          0.481 |
  | db7  | east        | slave | ONLINE |                4 |          0.451 |
  ```
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 [263], --roles [263]**

Limit the output to show only the specified role or comma-separated list of roles:

```
shell> multi_trepctl --roles=slave
```

- **--service [263], --services [263]**

Limit the output to the specified service or comma-separated list of services:

```
shell> multi_trepctl --service=east
```

- **--skip-headers [263]**

Prevents the generation of the headers when generating the list output format:

```
shell> multi_trepctl --skip-headers
```

- **--sort-by [263]**

Sort by the specified fieldname. For example, to sort the output by the latency:

```
shell> multi_trepctl --sort-by appliedlatency
```

### 8.11.2. 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>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>backups</td>
<td>List all the backups available across all configured hosts and services</td>
</tr>
<tr>
<td>heartbeat</td>
<td>Inserts a heartbeat on all masters within the service</td>
</tr>
<tr>
<td>list</td>
<td>List the information about each service</td>
</tr>
<tr>
<td>masterof</td>
<td>List all the masters of configured hosts and services</td>
</tr>
<tr>
<td>run</td>
<td>Run the specified trepctl command on all hosts/services</td>
</tr>
</tbody>
</table>

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
```
8.11.2.1. 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 |
| host2 | alpha       | 2014-08-15 09:36:57 | store-0000000001 | mysqldump |
```

8.11.2.2. 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: 5277
appliedlatency: 0.476
output: |
```

8.11.2.3. multi_trepctl masterof Command

Lists which hosts are masters of others within the configured services.

```
shell> multi_trepctl masterof
| servicename | host  | uri               |
| alpha       | host1 | thl://host1:2112/ |
```

8.11.2.4. 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 list
| host  | servicename | role   | state                      | appliedlastseqno | appliedlatency |
| host1 | firstrep    | master | OFFLINE:ERROR              | -1               | -1.000        |
| host2 | firstrep    | slave  | GOING-ONLINE:SYNCHRONIZING | 5277             | 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             | 0.480          |
```

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.

8.11.2.5. 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-8800000085.properties
```

```
host: host2
servicename: firstrep
output: |
  Backup completed successfully; URI=storage://file-system/store-8800000081.properties
```

```
host: host3
```
Return from the command will only take place when remote commands on each host have completed and returned.

8.12. The query Command

Table 8.13. query Common Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-conf PATH</td>
<td>Configuration file that contains values for connection properties (url, user and password)</td>
</tr>
<tr>
<td>-file PATH</td>
<td>File containing the SQL commands to run. If missing, read SQL commands from STDIN</td>
</tr>
<tr>
<td>-password</td>
<td>Prompt for password</td>
</tr>
<tr>
<td>-url JDBCURL</td>
<td>JDBC url of the database to connect to</td>
</tr>
<tr>
<td>-user USER</td>
<td>User used to connect to the database</td>
</tr>
</tbody>
</table>

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
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
  }
]
```

8.13. The replicator Command

The replicator is the wrapper script that handles the execution of the replicator service.

Table 8.14. replicator Commands

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>condrestart</td>
<td>Restart only if already running</td>
</tr>
<tr>
<td>console</td>
<td>Launch in the current console [instead of a daemon]</td>
</tr>
<tr>
<td>dump</td>
<td>Request a Java thread dump [if replicator is running]</td>
</tr>
<tr>
<td>install</td>
<td>Install the service to automatically start when the system boots</td>
</tr>
<tr>
<td>remove</td>
<td>Remove the service from starting during boot</td>
</tr>
<tr>
<td>restart</td>
<td>Stop replicator if already running and then start</td>
</tr>
</tbody>
</table>
Command-line Tools

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>Start in the background as a daemon process</td>
</tr>
<tr>
<td>status</td>
<td>Query the current status</td>
</tr>
<tr>
<td>stop</td>
<td>Stop if running (whether as a daemon or in another console)</td>
</tr>
</tbody>
</table>

These commands and options are described below:

- **condrestart** [266]

  **Table 8.15. Replicator Commands Options for condrestart [266]**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>offline</td>
<td>Start in OFFLINE state</td>
</tr>
</tbody>
</table>

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:

```bash
shell> replicator condrestart
Stopping Tungsten Replicator Service...
Waiting for Tungsten Replicator Service to exit...
Stopped Tungsten Replicator Service...
Starting Tungsten Replicator Service......
running: PID:26646
```

However, if not already running, the operation does nothing:

```bash
shell> replicator condrestart
Stopping Tungsten Replicator Service...
Tungsten Replicator Service was not running.
```

- **console** [266]

  **Table 8.16. Replicator Commands Options for console [266]**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>offline</td>
<td>Start in OFFLINE state</td>
</tr>
</tbody>
</table>

Launch in the current console (instead of a daemon)

- **dump** [266]

  Request a Java thread dump (if replicator is running)

- **install** [266]

  Installs the startup scripts for running the replicator at boot. For an alternative method of deploying these start-up scripts, see deployall.

- **remove** [266]

  Removes the startup scripts for running the replicator at boot. For an alternative method of removing these start-up scripts, see undeployall.

- **restart** [266]

  **Table 8.17. Replicator Commands Options for restart [266]**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>offline</td>
<td>Stop and restart in OFFLINE state</td>
</tr>
</tbody>
</table>

**Warning**

Restarting a running replicator temporarily stops and restarts replication.

Stops the replicator, if it is already running, and then restarts it:

```bash
shell> replicator restart
```
**Command-line Tools**

Stopping Tungsten Replicator Service...
Stopped Tungsten Replicator Service.
Starting Tungsten Replicator Service...
waiting for Tungsten Replicator Service......
running: PID:26248

- `start` [267]

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>offline</td>
<td>Start in OFFLINE state</td>
</tr>
</tbody>
</table>

Table 8.18. **replicator** Commands Options for `start` [267]

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

```shell
cd /path/to/replicator
replicator start
```

- `status` [267]

Checks the execution status of the replicator:

```shell
cd /path/to/replicator
replicator status
```

If the replicator is not running:

```shell
cd /path/to/replicator
replicator status
```

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` [267]

Stops the replicator if it is already running:

```shell
cd /path/to/replicator
replicator stop
```

8.14. The **setupCDC.sh** Command

The `setupCDC.sh` script configures an Oracle database with the necessary CDC tables to enable heterogeneous replication from Oracle to MySQL.

The script accepts one argument, the filename of the configuration file that will define the CDC configuration. The file accepts the parameters as listed in Table 8.19, "**setupCDC.conf Configuration Options**".

Table 8.19. **setupCDC.conf** Configuration Options

<table>
<thead>
<tr>
<th>CmdLine Option</th>
<th>INI File Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cdc_type [268]</td>
<td>cdc_type [268]</td>
<td>The CDC type to be used to extract data, either synchronous [using triggers] or asynchronous [using log processing].</td>
</tr>
<tr>
<td>delete_publisher</td>
<td>delete_publisher</td>
<td>Whether the publisher user should be deleted.</td>
</tr>
<tr>
<td>delete_subscriber</td>
<td>delete_subscriber</td>
<td>Whether the subscriber user should be deleted.</td>
</tr>
<tr>
<td>enable_archivelog</td>
<td>enable_archivelog</td>
<td>If set to true, the Oracle instance will be configured to enable archive logging [required by Oracle CDC], and then the Oracle instance will be restarted.</td>
</tr>
<tr>
<td>pub_password</td>
<td>pub_password</td>
<td>The publisher password that will be created for the CDC service.</td>
</tr>
<tr>
<td>pub_tablespace</td>
<td>pub_tablespace</td>
<td>Use the specified tablespace for CDC publishing instead of system tablespace</td>
</tr>
<tr>
<td>pub_user</td>
<td>pub_user</td>
<td>The publisher user that will be created for this CDC service.</td>
</tr>
<tr>
<td>service</td>
<td>service</td>
<td>The service name of the Tungsten Replicator service that will be created.</td>
</tr>
</tbody>
</table>
### Command-line Tools

<table>
<thead>
<tr>
<th>CommandLine Option</th>
<th>INI File Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>source_user</td>
<td>source_user</td>
<td>The source schema user with rights to access the database.</td>
</tr>
<tr>
<td>specific_path</td>
<td>specific_path</td>
<td>The path where the tungsten.tables file is located; the file must be in a shared location accessible by Tungsten Replicator.</td>
</tr>
<tr>
<td>specific_tables</td>
<td>specific_tables</td>
<td>If enabled, extract only the tables defined within a tungsten.tables file.</td>
</tr>
<tr>
<td>sys_pass</td>
<td>sys_pass</td>
<td>The system password to connect to Oracle as SYSDBA.</td>
</tr>
<tr>
<td>sys_user [270]</td>
<td>sys_user [270]</td>
<td>The system user to connect to Oracle as SYSDBA.</td>
</tr>
<tr>
<td>tungsten_pwd</td>
<td>tungsten_pwd</td>
<td>The password for the subscriber user.</td>
</tr>
<tr>
<td>tungsten_user</td>
<td>tungsten_user</td>
<td>The subscriber (Tungsten user) that will subscribe to the changes and read the information from the CDC tables.</td>
</tr>
</tbody>
</table>

Where:

- **cdc_type** [268]

<table>
<thead>
<tr>
<th>Option</th>
<th>Config File Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cdc_type [268]</td>
<td>cdc_type [268]</td>
<td>The CDC type to be used to extract data, either synchronous (using triggers) or asynchronous (using log processing).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value Type</th>
<th>Valid Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>CDCASYNC</td>
<td>Enable asynchronous capture</td>
</tr>
<tr>
<td></td>
<td>CDCSYNC</td>
<td>Enable synchronous capture</td>
</tr>
</tbody>
</table>

The CDC type to be used to extract data, either synchronous (using triggers) or asynchronous (using log processing).

- **delete_publisher**

<table>
<thead>
<tr>
<th>Option</th>
<th>Config File Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>delete_publisher</td>
<td>delete_publisher</td>
<td>Whether the publisher user should be deleted.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value Type</th>
<th>Valid Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>0</td>
<td>Do not the delete the user before creation</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Delete the user before creation</td>
</tr>
</tbody>
</table>

Whether the publisher user should be deleted.

- **deleteSubscriber**

<table>
<thead>
<tr>
<th>Option</th>
<th>Config File Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>deleteSubscriber</td>
<td>deleteSubscriber</td>
<td>Whether the subscriber user should be deleted.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value Type</th>
<th>Valid Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>0</td>
<td>Do not the delete the user before creation</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Delete the user before creation</td>
</tr>
</tbody>
</table>

Whether the subscriber user should be deleted.

- **enable_archivelog**

<table>
<thead>
<tr>
<th>Option</th>
<th>Config File Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable_archivelog</td>
<td>enable_archivelog</td>
<td>If set to true, the Oracle instance will be configured to enable archive logging (required by Oracle CDC), and then the Oracle instance will be restarted.</td>
</tr>
</tbody>
</table>
### Value Type

**boolean**

<table>
<thead>
<tr>
<th>Valid Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Do not automatically enable the archive log during setup</td>
</tr>
<tr>
<td>1</td>
<td>Automatically enable the archive log during setup</td>
</tr>
</tbody>
</table>

If set to true, the Oracle instance will be configured to enable archive logging (required by Oracle CDC), and then the Oracle instance will be restarted.

#### pub_password

<table>
<thead>
<tr>
<th>Option</th>
<th>pub_password</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>pub_password</td>
</tr>
<tr>
<td>Description</td>
<td>The publisher password that will be created for the CDC service.</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

The publisher password that will be created for the CDC service.

#### pub_tablespace

<table>
<thead>
<tr>
<th>Option</th>
<th>pub_tablespace</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>pub_tablespace</td>
</tr>
<tr>
<td>Description</td>
<td>Use the specified tablespace for CDC publishing instead of system tablespace</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

By default, the system tablespace is used for holding the publisher tables. Using the system tablespace should only be used during testing, as the tablespace is typically not large enough to hold the required change data. If set to 1, use the created tablespace (matching the value of `pub_user`) which is assumed to be large enough to hold the change information.

#### pub_user

<table>
<thead>
<tr>
<th>Option</th>
<th>pub_user</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>pub_user</td>
</tr>
<tr>
<td>Description</td>
<td>The publisher user that will be created for this CDC service.</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

The publisher user that will be created for this CDC service.

#### service

<table>
<thead>
<tr>
<th>Option</th>
<th>service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>service</td>
</tr>
<tr>
<td>Description</td>
<td>The service name of the Tungsten Replicator service that will be created.</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

The service name of the Tungsten Replicator service that will be created.

#### source_user

<table>
<thead>
<tr>
<th>Option</th>
<th>source_user</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>source_user</td>
</tr>
<tr>
<td>Description</td>
<td>The source schema user with rights to access the database.</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

The source schema user with rights to access the database.
## Command-line Tools

<table>
<thead>
<tr>
<th>Option</th>
<th>Config File Options</th>
<th>Description</th>
<th>Value Type</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>specific_path</td>
<td>specific_path</td>
<td>The path where the tungsten.tables file is located; the file must be in a shared location accessible by Tungsten Replicator.</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>specific_tables</td>
<td>specific_tables</td>
<td>If enabled, extract only the tables defined within a tungsten.tables file.</td>
<td>string</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>sys_pass</td>
<td>sys_pass</td>
<td>The system password to connect to Oracle as SYSDBA.</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>sys_user</td>
<td>sys_user [270]</td>
<td>The system user to connect to Oracle as SYSDBA.</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>tungsten_pwd</td>
<td>tungsten_pwd</td>
<td>The password for the subscriber user.</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>tungsten_user</td>
<td>tungsten_user</td>
<td>The subscriber (Tungsten user) that will subscribe to the changes and read the information from the CDC tables.</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

To use, supply the name of the configuration file to `setupCDC.sh`:
8.15. The startall Command

The `startall` command will start all configured services within the configured directory:

```
shell> ./setupCDC.sh sample.conf
```

- **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-5.4.1-41_pid25898/tungsten-connector/bin/../var/tconnector.pid
```

8.16. 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.**

8.17. 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 8.20, “thl Options”.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-conf path</code></td>
<td>Path to the configuration file containing the required replicator service configuration</td>
</tr>
<tr>
<td><code>-service servicename</code></td>
<td>Name of the service to be used when looking for THL information</td>
</tr>
</tbody>
</table>

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:

- **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.

8.17.1. 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 #** [271]

  Valid for: `thl list`
Output the TRL 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 slave has failed, the replication data for that sequence number can be individually viewed. For example:

```
shell> thl list -seqno 15
SEQ# = 15 / FRAG# = 0 (last frag)
- FILE = thl.data.000000000001
- 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, 
  unique_checks = 0, sql_mode = 'NO_AUTO_VALUE_ON_ZERO',
  service=firstrep,shard=cheffy]
- 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 TRL as their negative equivalents, and translated to the correct unsigned type when the data is applied to the target database.

- `-low [272]` and/or `-high [272]`
- `-from [272]` and/or `-to [272]`

Valid for: `thl list`, `thl purge`

Specify the start (`-low [272]`) or end (`-high [272]`) of the range of sequence numbers to be output. If only `-low [272]` is specified, then all sequence numbers from that number to the end of the TRL are output. If `-high [272]` 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.

5.1.0. Starting in 5.1.0, the options `-from [272]` and `-to [272]` can be used in place of `-low [272]` and `-high [272]` respectively.

For example:

```
shell> thl list -low 320
```

Or:

```
shell> thl list -from 320
```

Will output all the sequence number fragments from number 320.

```
shell> thl list -high 540
```

Or:

```
shell> thl list -high 540
```

Will output all the sequence number fragments up to and including 540.

```
shell> thl list -low 320 -high 540
```

Or:

```
shell> thl list -from 320 -to 540
```

Will output all the sequence number fragments from number 320 up to, and including, sequence number 540.

- `-first [272]`

Valid for: `thl list`, `thl purge`
5.2.0. Starting in 5.2.0, the options \texttt{-first}, \texttt{-first #}, \texttt{-last}, and \texttt{-last #}, can be used to select THL events.

The \texttt{-first} selects only the first stored THL event. For example:

\begin{verbatim}
shell> thl list -first
SEQ# = 0 / FRAG# = 0 (last frag)
- TIME = 2017-06-28 13:12:38.0
- EPOCH# = 0
...
\end{verbatim}

\textbf{• \texttt{-first #}}

Valid for: \texttt{thl list, thl purge}

5.2.0. Starting in 5.2.0, the options \texttt{-first}, \texttt{-first #}, \texttt{-last}, and \texttt{-last #}, can be used to select THL events.

The \texttt{-first #} selects the specified number of events, starting from the first event. For example:

\begin{verbatim}
shell> thl list -first 5

Would display the first five events from the stored THL.
\end{verbatim}

\textbf{• \texttt{-last}}

Valid for: \texttt{thl list, thl purge}

5.2.0. Starting in 5.2.0, the options \texttt{-first}, \texttt{-first #}, \texttt{-last}, and \texttt{-last #}, can be used to select THL events.

The \texttt{-last} selects only the last stored THL event. For example:

\begin{verbatim}
shell> thl list -last
SEQ# = 1601 / FRAG# = 0 (last frag)
- TIME = 2017-06-29 06:02:23.0
- EPOCH# = 1601
...
\end{verbatim}

The use of this option can be particularly useful in the event of synchronisation or THL corruption due to a lack of disk space. Using the \texttt{thl purge} command, the last THL event can be easily removed without having to work out the ranges and index information:

\begin{verbatim}
shell> thl purge -last
\end{verbatim}

\textbf{• \texttt{-last #}}

Valid for: \texttt{thl list, thl purge}

5.2.0. Starting in 5.2.0, the options \texttt{-first}, \texttt{-first #}, \texttt{-last}, and \texttt{-last #}, can be used to select THL events.

The \texttt{-last #} selects the specified number of events, starting from the last event. For example:

\begin{verbatim}
shell> thl list -last 5

When the THL index contains events from 1558-1601, would display events 1597 through to 1601.
\end{verbatim}

8.17.2. \texttt{thl list} Command

The \texttt{list} command to the \texttt{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.

\texttt{thl list}


\textbf{• \texttt{-file filename}}

Outputs all of the sequence number fragment information from the specified THL file. If the filename has been determined from the \texttt{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.

\textbf{• \texttt{-charset charset}}

273
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 [274]**

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 [274]**

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 2013-05-03 20:58:14.0 mysql-bin.000005:0000000579721045;0 host3
2047 1412 1 true 2013-05-03 20:58:14.0 mysql-bin.000005:0000000579721116;0 host3
2048 1412 0 false 2013-05-03 20:58:14.0 mysql-bin.000005:0000000580759286;0 host3
2048 1412 1 true 2013-05-03 20:58:14.0 mysql-bin.000005:0000000580759277;0 host3
2049 1412 0 false 2013-05-03 20:58:16.0 mysql-bin.000005:0000000581791468;0 host3
2049 1412 1 true 2013-05-03 20:58:16.0 mysql-bin.000005:0000000581791539;0 host3
2050 1412 0 false 2013-05-03 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 these sequence objects. For example:

```
[
  {
    "lastFrag" : false,
    "epoch" : 7,
    "seqno" : 320,
    "time" : "2013-05-02 11:41:19.0",
    "frag" : 0,
    "comments" : "",
    "sourceId" : "host1",
    "eventId" : "mysql-bin.000004:0000000244498614;0"
  },
  {
    "lastFrag" : true,
    "epoch" : 7,
    "seqno" : 320,
    "time" : "2013-05-02 11:41:19.0",
    "frag" : 1,
    "comments" : "",
    "sourceId" : "host1",
    "eventId" : "mysql-bin.000004:0000000244498685;0"
  }
]
```

For more information on the fields displayed, see THL SEQNO [503].

- **-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
```
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` and/or `-last`.

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 provides 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 2017-06-29 11:04:53.0 123 chunks SQL 0 bytes (0 avg bytes per chunk) Rows 45633 (371 avg rows per chunk)
  1604 1 2017-06-29 11:04:53.0 123 chunks SQL 0 bytes (0 avg bytes per chunk) Rows 45633 (371 avg rows per chunk)
  1604 2 2017-06-29 11:04:53.0 123 chunks SQL 0 bytes (0 avg bytes per chunk) Rows 45633 (371 avg rows per chunk)
  1604 3 2017-06-29 11:04:53.0 123 chunks SQL 0 bytes (0 avg bytes per chunk) Rows 45633 (371 avg rows per chunk)
  1604 4 2017-06-29 11:04:53.0 123 chunks SQL 0 bytes (0 avg bytes per chunk) Rows 45633 (371 avg rows per chunk)
  1604 5 2017-06-29 11:04:53.0 123 chunks SQL 0 bytes (0 avg bytes per chunk) Rows 45633 (371 avg rows per chunk)
  1604 6 2017-06-29 11:04:53.0 123 chunks SQL 0 bytes (0 avg bytes per chunk) Rows 45633 (371 avg rows per chunk)
  1604 7 2017-06-29 11:04:53.0 123 chunks SQL 0 bytes (0 avg bytes per chunk) Rows 45633 (371 avg rows per chunk)
  Event total: 1360 chunks 0 bytes in SQL statements 504498 rows
  ```

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` and/or `-last`.

- `sizessummary`

  Outputs only the size summary information for the requested THL:

  ```
  shell> thl list -sizes -summary
  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 = 2014-01-30 05:46:26.0
  - EPOCH = 5278
  - EVENTID = mysql-bin.000017:0000000000001117;0
  - SOURCEID = host1
  - METADATA = [mysql_server_id=1687811;dbms_type=mysql;is_metadata=true;]
    service_firststep,heartbeat=MASTER_ONLINE]
  - TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
  - SQL(0) =
    - ACTION = UPDATE
    - SCHEMA = tungsten_firststep
    - TABLE = heartbeat
    - ROW = 0
  ```

275
8.17.3. 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:

```bash
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` [274] ignores the checksum information on events in the event that the checksum is corrupt.

8.17.4. thl purge Command

The purge command to the thl command deletes sequence number information from the THL files.

```bash
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 master 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:

```bash
shell> thl index
```
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 \texttt{-high} to specify the highest number to be removed (6021):

\begin{verbatim}
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
2017-02-10 16:31:36,235 [ - main] INFO  thl.THLManagerCtrl Transactions deleted
Running a \texttt{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)

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)
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 \texttt{-low} to specify the sequence number:

\begin{verbatim}
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
2017-02-10 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)
LogIndexEntry thl.data.0000000399(6242:6261)
LogIndexEntry thl.data.0000000400(6262:6266)

The confirmation message can be bypassed by using the \texttt{-y} option, which implies that the operation should proceed without further confirmation.

The optional argument \texttt{-no-checksum} 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` 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.

### 8.17.5. `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, 2013-12-18 11:53:00)
newest file = thl.data.0000000041 (0.98 MB, 2013-12-18 12:34:32)
```

The optional argument `-no-checksum` ignores the checksum information on events in the event that the checksum is corrupt.

### 8.17.6. `thl help` Command

The `help` command to the `thl` command outputs the current help message text.

### 8.18. 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
- 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 8.18.1, "trepctl Options".](#)
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 8.18.2, "trepctl Global Commands". Information on individual commands can be found in Section 8.18.3, "trepctl Service Commands".

8.18.1. trepctl Options

Table 8.21. trepctl Command-line Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-host name</td>
<td>Host name of the replicator</td>
</tr>
<tr>
<td>-port number</td>
<td>Port number of the replicator</td>
</tr>
<tr>
<td>-retry N</td>
<td>Number of times to retry the connection</td>
</tr>
<tr>
<td>-service name</td>
<td>Name of the replicator service</td>
</tr>
<tr>
<td>-verbose</td>
<td>Enable verbose messages for operations</td>
</tr>
</tbody>
</table>

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.

```sh
trepctl status
Processing status command...
Operation failed: You must specify a service name with the -service flag
```

- **-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.

8.18.2. 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 8.22. trepctl Replicator Wide Commands

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kill</td>
<td>Shutdown the replication services immediately</td>
</tr>
<tr>
<td>services</td>
<td>List the configured replicator services</td>
</tr>
<tr>
<td>version</td>
<td>Show the replicator version number and build</td>
</tr>
</tbody>
</table>

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 not required.

8.18.2.1. 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
```

8.18.2.2. 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
```

280
The information can be reported in JSON format by using the \texttt{-json} option to the command:

{
  "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 \texttt{-full} option is added, the JSON output includes full details of the service, similar to that output by the \texttt{trepctl status} command, but for each configured service:

{
  "masterConnectUri": "",
  "rmPort": "10000",
  "masterListenUri": "thl://host1:2112/",
  "pendingErrorSeqno": "-1",
  "pipelineSource": "jdbc:mysql:thin://host1:3306/",
  "masterConnectUri": "",
  "appliedLastSeqno": "2541",
  "pendingExceptionMessage": "NONE",
  "simpleServiceName": "alpha",
  "appliedLastSeqno": "2541",
  "pendingError": "NONE",
  "pendingErrorEventId": "NONE",
  "started": "true",
  "siteName": "default",
  "pendingLatency": "245804.198",
  "relativeLatency": "245804.198",
  "role": "master",
  "serviceName": "alpha",
  "pendingErrorCode": "NONE",
  "masterListenUri": "thl://host1:2112/",
  "pendingErrorSeqno": "-1",
  "pipelineSource": "jdbc:mysql:thin://host1:3306/",
  "masterConnectUri": "",
  "appliedLastSeqno": "2541",
  "pendingExceptionMessage": "NONE",
  "simpleServiceName": "alpha",
  "appliedLastSeqno": "2541",
  "pendingError": "NONE",
  "pendingErrorEventId": "NONE",
  "started": "true",
  "siteName": "default",
  "pendingLatency": "245804.198",
  "relativeLatency": "245804.198",
  "role": "master",
  "serviceName": "alpha",
  "pendingErrorCode": "NONE",
  "masterConnectUri": "",
  "appliedLastSeqno": "2541",
  "pendingExceptionMessage": "NONE",
  "simpleServiceName": "alpha",
  "appliedLastSeqno": "2541",
  "pendingError": "NONE",
  "pendingErrorEventId": "NONE",
  "started": "true",
  "siteName": "default",
  "pendingLatency": "245804.198",
  "relativeLatency": "245804.198",
  "role": "master",
  "serviceName": "alpha",
  "pendingErrorCode": "NONE",
  "masterConnectUri": "",
  "appliedLastSeqno": "2541",
  "pendingExceptionMessage": "NONE",
  "simpleServiceName": "alpha",
  "appliedLastSeqno": "2541",
  "pendingError": "NONE",
  "pendingErrorEventId": "NONE",
  "started": "true",
  "siteName": "default",
  "pendingLatency": "245804.198",
  "relativeLatency": "245804.198",
  "role": "master",
  "serviceName": "alpha",
  "pendingErrorCode": "NONE",
  "masterConnectUri": "",
  "appliedLastSeqno": "2541",
  "pendingExceptionMessage": "NONE",
  "simpleServiceName": "alpha",
  "appliedLastSeqno": "2541",
  "pendingError": "NONE",
  "pendingErrorEventId": "NONE",
  "started": "true",
  "siteName": "default",
  "pendingLatency": "245804.198",
  "relativeLatency": "245804.198",
  "role": "master",
  "serviceName": "alpha",
  "pendingErrorCode": "NONE",
  "masterConnectUri": "",
  "appliedLastSeqno": "2541",
  "pendingExceptionMessage": "NONE",
  "simpleServiceName": "alpha",
  "appliedLastSeqno": "2541",
  "pendingError": "NONE",
  "pendingErrorEventId": "NONE",
  "started": "true",
  "siteName": "default",
  "pendingLatency": "245804.198",
  "relativeLatency": "245804.198",
  "role": "master",
  "serviceName": "alpha",
  "pendingErrorCode": "NONE",
  "masterConnectUri": "",
  "appliedLastSeqno": "2541",
  "pendingExceptionMessage": "NONE",
  "simpleServiceName": "alpha",
  "appliedLastSeqno": "2541",
  "pendingError": "NONE",
  "pendingErrorEventId": "NONE", 281
Auto-refresh support added in 6.0.1. Starting with Tungsten Clustering 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".

### 8.18.2.3. `trepctl version` Command

The `trepctl version` command outputs the version number of the specified replicator service.

```
trepctl version
shell> trepcctl version
Tungsten Replicator 5.4.1 build 41
```

The system can also be used to obtain remote version:

```
trepctl -host host2 version
shell> trepcctl -host host2 version
Tungsten Replicator 5.4.1 build 41
```

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.

### 8.18.3. `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 8.23. `trepctl` Service Commands**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>backup</td>
<td>Backup database</td>
</tr>
<tr>
<td>capabilities</td>
<td>List the configured replicator capabilities</td>
</tr>
<tr>
<td>check</td>
<td>Generate consistency check</td>
</tr>
<tr>
<td>clear</td>
<td>Clear one or all dynamic variables</td>
</tr>
<tr>
<td>clients</td>
<td>List clients connected to this replicator</td>
</tr>
<tr>
<td>flush</td>
<td>Synchronize transaction history log to database</td>
</tr>
<tr>
<td>heartbeat</td>
<td>Insert a heartbeat event with optional name</td>
</tr>
<tr>
<td>load</td>
<td>Load the replication service</td>
</tr>
<tr>
<td>offline</td>
<td>Set replicator to OFFLINE state</td>
</tr>
<tr>
<td>offline-deferred</td>
<td>Set replicator OFFLINE at a future point in the replication stream</td>
</tr>
<tr>
<td>online</td>
<td>Set Replicator to ONLINE with start and stop points</td>
</tr>
<tr>
<td>perf</td>
<td>Print detailed performance information</td>
</tr>
<tr>
<td>properties</td>
<td>Display a list of all internal properties</td>
</tr>
<tr>
<td>purge</td>
<td>Purge non-Tungsten logins on database</td>
</tr>
<tr>
<td>qs</td>
<td>Print a simplified quick replicator status</td>
</tr>
<tr>
<td>reset</td>
<td>Deletes the replicator service</td>
</tr>
<tr>
<td>restore</td>
<td>Restore database on specified host</td>
</tr>
<tr>
<td>setrole</td>
<td>Set replicator role</td>
</tr>
<tr>
<td>shard</td>
<td>List, add, update, and delete shards</td>
</tr>
<tr>
<td>status</td>
<td>Print replicator status information</td>
</tr>
<tr>
<td>unload</td>
<td>Unload the replication service</td>
</tr>
<tr>
<td>wait</td>
<td>Wait for the replicator to reach a specific state, time or applied sequence number</td>
</tr>
</tbody>
</table>
The following sections detail each command individually, with specific options, operations and information.

### 8.18.3.1. trepctl backup Command

Tungsten Clustering 5.0.0. This feature has been deprecated in Tungsten Clustering 5.0.0 and will be removed in a future release.

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 8.24. trepctl backup Command Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-backup agent</code></td>
<td>Select the backup agent</td>
</tr>
<tr>
<td><code>-limit s</code></td>
<td>The period to wait before returning after the backup request</td>
</tr>
<tr>
<td><code>-storage agent</code></td>
<td>Select the storage agent</td>
</tr>
</tbody>
</table>

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 8.18.3.17, "trepctl restore Command". Different backup solutions may require that the replicator be placed into the `OFFLINE` state before the backup is performed.

A log of the backup operation will be stored in the replicator log directory, if 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` 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` 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` 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.

```
... BackupTask Storing backup result...
... backup.FileSystemStorageAgent Allocated backup location: »
... 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
... BackupTask Backup completed normally: »
uri=storage://file-system/store-8888888881.properties
```

The URI can be used during a restore.

### 8.18.3.2. 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 slaves 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 8.18.3.3, "trepctl check Command".

- **Heartbeat**
  Indicates whether the heartbeat service is supported. For more information see Section 8.18.3.7, "trepctl heartbeat Command".

- **Flush**
  Indicates whether the trepctl flush operation is supported.

### 8.18.3.3. 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 master and the slave.

**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.

### 8.18.3.4. 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 master in place of a slave. The replicator can be returned to the initial configuration for the service by clearing this dynamic property:

```
shell> trepctl clear
```

### 8.18.3.5. trepctl clients Command

Outputs a list of the that have been connected to the master service since it went online. If a slave service goes offline or is stopped, it will still be reported by this command.

```
trepctl clients [-json]
```

Where:
Table 8.25. `trepctl clients` Command Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-json</code></td>
<td>Output the information as JSON</td>
</tr>
</tbody>
</table>

The command outputs the list of clients and the management port on which they can be reached:

```shell
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` option:

```shell
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.

8.18.3.6. `trepctl flush` Command

On a master, 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 slave.

```
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
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
shell> trepctl flush -limit 1
```

8.18.3.7. `trepctl heartbeat` Command

Inserts a heartbeat into the replication stream, which can be used to identify replication points.

```
trepctl heartbeat [-name ]
```

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
shell> trepctl heartbeat
```

When performing specific operations, the heartbeat can be given an name:

```shell
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 slave. 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 slave without requiring changes to the database. To check replication using the heartbeat:
1. Check the current transaction sequence number on the master:

```shell
shell> trepctl status
Processing status command...
NAME                     VALUE
....                     .....  
appliedLastEventId     : mysql-bin.000009:0000000000008998;0
appliedLastSeqno       : 3630
...
```

2. Insert a heartbeat event:

```shell
shell> trepctl heartbeat
```

3. Check the sequence number again:

```shell
shell> trepctl status
Processing status command...
NAME                     VALUE
....                     .....  
appliedLastEventId     : mysql-bin.000009:0000000000009310;0
appliedLastSeqno       : 3631
```

4. Check that the sequence number on the slave matches:

```shell
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 slave before enabling standard replication. This can be achieved by configuring the slave to go offline when a specific heartbeat event is seen, loading the data on the master, inserting the heartbeat when the load has finished, and then performing the slave backup:

1. On the slave:

```shell
slave shell> trepctl offline-deferred -at-heartbeat dataload
```

The `trepctl offline-deferred` configures the slave 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:

```shell
NAME                     VALUE
....                     .....  
appliedLastEventId     : mysql-bin.000009:0000000000009310;0
appliedLastSeqno       : 3631
appliedLatency         : 0.704
...
offlineRequests        : Offline at heartbeat event: dataload
```

2. On the master:

```shell
master shell> mysql newdb < newdb.load
```

3. Once the data load has completed, insert the heartbeat on the master:

```shell
master 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 slave will go into the offline state. Now a backup can be created with all of the loaded data replicated from the master. Because the slave is in the offline state, no further data or changes will be recorded on the slave.

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.

### 8.18.3.7.1. 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)
```
During replication, slaves 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.

### 8.18.3.8. trepctl load Command

Load the replicator service.

```shell
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
```

### 8.18.3.9. trepctl offline Command

The `trepctl offline` command puts the replicator into the offline state, stopping replication.

```shell
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
```

### 8.18.3.10. 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.

```shell
trepctl offline-deferred [-at-event event] [-at-heartbeat [heartbeat]] [-at-seqno seqno] [-at-time YYYY-MM-DD_hh:mm:ss]
```

Where:

**Table 8.26. trepctl offline-deferred Command Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-at-event event</code> [288]</td>
<td>Go offline at the specified event</td>
</tr>
<tr>
<td><code>-at-heartbeat [heartbeat]</code> [288]</td>
<td>Go offline when the specified heartbeat is identified</td>
</tr>
</tbody>
</table>
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 8.18.3.9, "trepctl offline Command".

The supported triggers are:

- **-at-seqno [288]**
  
  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 [288]**
  
  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 [288]**
  
  Specifies the name of a specific heartbeat to look for when replication should be stopped.

- **-at-time [288]**
  
  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.

### 8.18.3.11. 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.

Where:

Table 8.27. trepctl online Command Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-all-services</td>
<td>Place online all available services</td>
</tr>
<tr>
<td>-base-seqno x</td>
<td>On a master, restart replication using the specified sequence number</td>
</tr>
<tr>
<td>-force</td>
<td>Force the online state</td>
</tr>
<tr>
<td>-from-event event</td>
<td>Start replication from the specified event</td>
</tr>
<tr>
<td>-no-checksum</td>
<td>Disable checksums for all events when going online</td>
</tr>
<tr>
<td>-provision [SCN]</td>
<td>Start provisioning using the parallel extractor</td>
</tr>
<tr>
<td>-skip-seqno seqdef</td>
<td>Skip one, multiple, or ranges of sequence numbers before going online</td>
</tr>
<tr>
<td>-until-event event</td>
<td>Define an event when replication will stop</td>
</tr>
<tr>
<td>-until-heartbeat [name]</td>
<td>Define a heartbeat when replication will stop</td>
</tr>
<tr>
<td>-until-seqno seqno</td>
<td>Define a sequence no when replication will stop</td>
</tr>
<tr>
<td>-until-time YYYY-MM-DD_hh:mm:ss</td>
<td>Define a time when replication will stop</td>
</tr>
</tbody>
</table>

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.

### 8.18.3.11.1. Going Online from Specific Transaction Points

If there is one, or more, event in the THL that could not be applied to the slave 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
  ```

### 8.18.3.11.2. 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:
8.18.3.11.3. 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.

8.18.3.11.4. 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.

8.18.3.11.4.1. 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.

8.18.3.11.4.2. 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.

8.18.3.11.4.3. 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
```

8.18.3.11.4.4. 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.

### 8.18.3.11.5. 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
```

### 8.18.3.11.6. Going Online and Starting Provisioning

Supported only for Oracle Extraction. The `-provision` option is only supported for extraction from Oracle.

The replicator can be configured to go online and start populating the THL using information generated by the Section 7.11, “Using the Parallel Extractor” process. This generates THL information, inserting the data generated by the parallel extractor before starting to extract data from the source database. This effectively populates the THL (and downstream slave databases) with existing data from the source database.

To start provisioning, extracting all data:

```
shell> trepctl online -provision
```

To start provisioning, extracting data from the existing database using a specific reference point, such as the Oracle System Change Number (SCN), append the number to the example:

```
shell> trepctl online -provision 45987459
```

### 8.18.3.11.7. 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.

### 8.18.3.12. trepctl perf Command

Display a list of all the internal properties. The list can be filtered.

```
trepctl perf [ -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 | 68.173s | 8.109s | 0.815s | 0.004s | 68.381s
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 | 56.859s | 8.020s | 5.247s | 0.828s | 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 | 3.163s | 42 | 37.831s | 0.001s | 8.003s | 0.011s | 38.246s
Avg time per Event | 0.901s | 0.000s | 8.000s | 0.010s | 0.911s
q-to-dbms | 3235 | 3.746s | 164 | 22.226s | 0.019s | 15.242s | 0.338s | 37.825s
Avg time per Event | 0.014s | 0.000s | 8.009s | 0.009s | 0.023s
```

The individual statistics shown are as follows:
• All statistics within the replicator are reset when the replicator goes **ONLINE**. 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** number for the specified stage. A difference in sequence numbers is possible (as seen in the apply 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:

```
$ trepctl perf
Currently not online; performance stats not available
State: Safely Offline for 6.491s
```

### 8.18.13. 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:

```
$ trepctl properties
{
  "replicator.store.thl_log_file_retention": "7d",
  "replicator.filter.bidiSlave.allowBidilmsafe": "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:

```
$ 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",
  ...
}
```
8.18.3.14. **trepctl purge Command**

Forces all logins on the attached database, other than those directly related to Tungsten Clustering, to be disconnected. The command is only supported on master, and can be used to disconnect users before a switchover or taking a master offline to prevent further use of the system.

```
trepctl purge [ -limit s ] [ -y ]
```

Where:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-limit s</td>
<td>Specify the waiting time for the operation</td>
</tr>
<tr>
<td>-y</td>
<td>Indicates that the command should continue without interactive confirmation</td>
</tr>
</tbody>
</table>

**Warning**

Use of the command will disconnect running users and queries and may leave the database in 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 option:

```
sHELL> trepctl purge -y
Directing replicator to purge non-Tungsten sessions
Number of sessions purged: 0
```

An optional parameter, -wait, defines the period of time that the operation will wait before returning to the command-line.

An optional parameter, -limit, defines the period of time that the operation will wait before returning to the command-line.

8.18.3.15. **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:
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 master [extractor] the information shown describes the latency from extraction into the THL files:

- **Latency**: 1216.315s since last source commit
  - The next line shows the interval since the last time there was a database commit. On a master (extractor) is the time between the last database commit to the binary log and the information being written to THL. On a slave, 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

If the replicator is offline due to being deliberately placed offline using `trepctl offline` then the basic information and status is shown:

- **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
```

8.18.3.16. `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 8.29. `trepctl reset` Command Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-all [295]</td>
<td>Deletes the thl directory, relay logs directory and tungsten database for the service. Same as specifying -thl -relay -db</td>
</tr>
<tr>
<td>-db [295]</td>
<td>Deletes the tungsten_{service_name} database for the service</td>
</tr>
<tr>
<td>-relay [295]</td>
<td>Deletes the relay directory for the service</td>
</tr>
<tr>
<td>-thl [295]</td>
<td>Deletes the thl directory for the service</td>
</tr>
<tr>
<td>-y [295]</td>
<td>Indicates that the command should continue without interactive confirmation</td>
</tr>
</tbody>
</table>

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` 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.
- Reset the redo log contents of the service. Valid only for Oracle extraction deployments

### 8.18.3.17. `trepctl restore` Command

Tungsten Clustering 5.0.0. This feature has been deprecated in Tungsten Clustering 5.0.0 and will be removed in a future release.

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` state. The datasource should be switched `ONLINE` using `trepctl`:

```
shell> trepctl online
```

Any outstanding events from the master will be processed and applied to the slave, which will catch up to the current master status over time.

### 8.18.3.18. `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 slave and master roles, for example during switchover.
trepctl setrole [-role [master|relay|slave]] [ -uri ]

Where:

Table 8.30. trepctl setrole Command Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-role</td>
<td>Replicator role</td>
</tr>
<tr>
<td>-uri</td>
<td>URI of the master</td>
</tr>
</tbody>
</table>

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 slave, the URI of the master can be optionally supplied:

```shell
trepctl setrole -role slave -uri th://host1:2112/
```

8.18.3.19. trepctl shard Command

The `trepctl shard` command provides an interface to the replicator shard system definition system.

`trepctl shard [-delete shard] [-insert shard] [-list] [-update shard]`

Where:

Table 8.31. trepctl shard Command Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-delete shard</td>
<td>Delete a shard definition</td>
</tr>
<tr>
<td>-insert shard</td>
<td>Add a new shard definition</td>
</tr>
<tr>
<td>-list</td>
<td>List configured shards</td>
</tr>
<tr>
<td>-update shard</td>
<td>Update a shard definition</td>
</tr>
</tbody>
</table>

The replicator shard system is used during multi-site replication configurations to control where information is replicated.

8.18.3.19.1. Listing Current Shards

To obtain a list of the currently configured shards:

```shell
trepctl shard -list
```

The shard map information can also be captured and then edited to update existing configurations:

```shell
trepctl shard -list>shard.map
```

8.18.3.19.2. 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
```

8.18.3.19.3. Updating an Existing Shard Configuration

To update a definition:

```shell
trepctl shard -update < shard.map
Reading from standard input
```
8.18.3.19.4. Deleting a Shard Configuration

To delete a single shard definition, specify the shard name:

```shell
$ trepctl shard -delete alpha
```

8.18.3.20. `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 [-json] [-namechannel-assignmentsservicesshardstagesstorestaskswatches][[-r]
```

Where:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-json</code></td>
<td>Output the information in JSON format</td>
</tr>
<tr>
<td><code>-name</code></td>
<td>Select a specific group of status information</td>
</tr>
<tr>
<td><code>-r</code></td>
<td>Refresh the status display</td>
</tr>
</tbody>
</table>

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    : 1876.613
resourcePrecedence : 99
rmiPort            : 10098
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 5.4.1 build 41
```

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.
8.18.3.20.1. Getting Detailed Status

More detailed information about selected areas of the replicator status can be obtained by using the `-name` option.

8.18.3.20.1.1. 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)...
```

8.18.3.20.1.2. 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)...
```

8.18.3.20.1.3. 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 environment not configured with parallel apply, the shards output will just show a single entry.

8.18.3.20.1.4. 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
```

298
8.18.3.20.15. 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 master 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
disableChecksum : false
flushIntervalMillis : 0
discardListDelay : false
logConnectionTimeout : 28800
logDir : /opt/continuent/thl/alpha
logFileRetentionMillis : 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
```
8.18.3.20.1.6. 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  : 3.590
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.

8.18.3.20.1.7. 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 output of watches will show the stages waiting on the specific seqno.
The following example shows the use of `offline-deferred` and the subsequent resulting output from `watches`.

```shell
trepctl offline-deferred -at-seqno 234
trepctl status -name watches
```

Processing status command (watches)...

<table>
<thead>
<tr>
<th>NAME</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>cancel tasks</td>
</tr>
<tr>
<td>cancelled</td>
<td>false</td>
</tr>
<tr>
<td>committed</td>
<td>false</td>
</tr>
<tr>
<td>done</td>
<td>false</td>
</tr>
<tr>
<td>matched</td>
<td>[[0:false]]</td>
</tr>
<tr>
<td>predicate</td>
<td>SeqnoWatchPredicate seqno=234</td>
</tr>
<tr>
<td>stage</td>
<td>remote-to-thl</td>
</tr>
</tbody>
</table>

NAME   | VALUE |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>cancel tasks</td>
</tr>
<tr>
<td>cancelled</td>
<td>false</td>
</tr>
<tr>
<td>committed</td>
<td>false</td>
</tr>
<tr>
<td>done</td>
<td>false</td>
</tr>
<tr>
<td>matched</td>
<td>[[0:false]]</td>
</tr>
<tr>
<td>predicate</td>
<td>SeqnoWatchPredicate seqno=234</td>
</tr>
<tr>
<td>stage</td>
<td>thl-to-q</td>
</tr>
</tbody>
</table>

NAME   | VALUE |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>cancel tasks</td>
</tr>
<tr>
<td>cancelled</td>
<td>false</td>
</tr>
<tr>
<td>committed</td>
<td>false</td>
</tr>
<tr>
<td>done</td>
<td>false</td>
</tr>
<tr>
<td>matched</td>
<td>[[0:false]]</td>
</tr>
<tr>
<td>predicate</td>
<td>SeqnoWatchPredicate seqno=234</td>
</tr>
<tr>
<td>stage</td>
<td>q-to-dms</td>
</tr>
</tbody>
</table>

Finished status command (watches)...

8.18.3.20.2. 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**

```json
{
    "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",
    "singleServiceName": "alpha",
    "extensions": "",
    "pendingErrorEventId": "NONE",
    "sourceId": "host1",
    "serviceName": "alpha",
    "version": "Tungsten Replicator 5.4.1 build 41",
    "role": "master",
    "currentTimeMillis": "1369233410874",
    "rmiPort": "10000",
    "siteName": "default",
    "pendingErrorSeqno": "-1",
    "seqnoType": "java.lang.Long",
    "currentEventId": "mysql-bin.000007:0000000000001353",
    "relativeLatency": "0.53",
    "pipelineSource": "jdbc:mysql:thin://host1:3306/",
    "pendingErrorCode": "NONE",
    "maximumStoredSeqNo": "2504",
    "latestEpochNumber": "2500",
    "channels": "1",
    "appliedLastSeqno": "2504",
    "serviceType": "local",
    "storageType": "java.lang.Long",
    "appliedLatency": "0.33",
    "pendingLatency": "2125.873",
    "relativeLatency": "2125.372",
    "dataServerHost": "host1"
}
```
8.18.3.20.2.1. Detailed Status: Channel Assignments JSON Output

```shell
shell> trepctl status -name channel-assignments -json
[
  {
    "channel": "0",
    "shard_id": "cheffy"
  },
  {
    "channel": "0",
    "shard_id": "tungsten_alpha"
  }
]
```

8.18.3.20.2.2. Detailed Status: Services JSON Output

```shell
shell> trepctl status -name services -json
[
  {
    "totalAssignments": "2",
    "accessFailures": "0",
    "storeClass": "com.continuent.tungsten.replicator.channel.ChannelAssignmentService",
    "name": "channel-assignment",
    "maxChannel": "0"
  }
]
```

8.18.3.20.2.3. Detailed Status: Shards JSON Output

```shell
shell> trepctl status -name shards -json
[
  {
    "stage": "q-to-dbms",
    "appliedLastEventId": "mysql-bin.000007:0000000007224342:0",
    "appliedLatency": "63.899",
    "appliedLastSeqno": "2514",
    "eventCount": "16",
    "shardId": "cheffy"
  }
]
```

8.18.3.20.2.4. Detailed Status: Stages JSON Output

```shell
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": "10",
    "processedMinSeqno": "-1",
    "extractor.class": "com.continuent.tungsten.replicator.thl.THLStoreExtractor"
  },
  {
    "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-dbs",
    "extractor.name": "parallel-q-extractor",
    "filter.1.name": "pkey",
    "taskCount": "1",
    "committedMinSeqno": "2504",
    "blockCommitRowCount": "10",
    "processedMinSeqno": "-1",
    "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",
    "filter.0.name": "mysqlsessions",
    "processedMinSeqno": "-1"
  }
]```
8.18.3.20.2.5. Detailed Status: Stores JSON Output

```shell
trepctl status -name stores -json
```
```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"
}
```

8.18.3.20.2.6. Detailed Status: Tasks JSON Output

```shell
trepctl status -name tasks -json
```
```json
{
"filterTime": "8.0",
"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",
"cancelled": "false"
}
{
"filterTime": "8.0",
"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",
"cancelled": "false"
}
{
"filterTime": "8.263",
"stage": "q-to-dbms",
"currentLastFragno": "1",
"taskId": "0",
"currentLastSeqno": "2614",
"state": "apply",
"extractTime": "533.471",
"applyTime": "61.618",
"averageBlockSize": "1.160"
}
```
8.18.3.20.2.7. Detailed Status: Tasks JSON Output

shell> trepctl status -name watches -json

8.18.3.21. trepctl unload Command

Unload the replicator service.

\texttt{trepctl unload [ \textsc{-y} ]}

Unload the replicator service entirely. An interactive prompt is provided to confirm the shutdown:

\texttt{shell> trepctl unload}

Do you really want to unload replication service alpha? \texttt{[yes/NO]}

To disable the prompt, use the \texttt{-y} option:

\texttt{shell> trepctl unload -y}

The name of the service unloaded is provided for confirmation.

8.18.3.22. trepctl wait Command

The \texttt{trepctl wait} command waits for the replicator to enter a specific state, or for a specific sequence number to be applied to the dataserver.

\texttt{trepctl wait [ \texttt{-applied seqno \[304\]} [ \texttt{-limit s \[304\]} [ \texttt{-state st \[304\]} ] ] ]}

Where:

\textbf{Table 8.33. trepctl wait Command Options}

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{-applied seqno [304]}</td>
<td>Specify the sequence number to be waited for</td>
</tr>
<tr>
<td>\texttt{-limit s [304]}</td>
<td>Specify the number of seconds to wait for the operation to complete</td>
</tr>
<tr>
<td>\texttt{-state st [304]}</td>
<td>Specify a state to be waited for</td>
</tr>
</tbody>
</table>

The command will wait for the specified occurrence, of either a change in the replicator status (i.e. \texttt{ONLINE \[212\]}), or for a specific sequence number to be applied. For example, to wait for the replicator to go into the \texttt{ONLINE \[212\]} state:

\texttt{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, \texttt{trepctl} returns with exit status 0.

To wait a specific sequence number to be applied:

\texttt{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 master to be applied on the slave. Unlike the \texttt{offline-deferred} operation, no change in the replicator is made. Instead, \texttt{trepctl} simply returns with exit status 0 when the sequence number has been successfully applied.

If the optional \texttt{-limit \[304\]} option is used, then \texttt{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:

\texttt{shell> trepctl wait -state ONLINE -limit 10}

If the requested event does not take place before the specified time limit expires, then \texttt{trepctl} returns with the message 'Wait timed out!', and an exit status of 1.
8.19. The `tpassword` Command

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--create, -c</td>
<td>Creates a new user/password</td>
</tr>
<tr>
<td>--delete, -d</td>
<td>Delete a user/password combination</td>
</tr>
<tr>
<td>-e, --encrypted.password</td>
<td>Encrypt the password</td>
</tr>
<tr>
<td>-f, --file</td>
<td>Specify the location of the security.properties file</td>
</tr>
<tr>
<td>-h, --help</td>
<td>Display help text</td>
</tr>
<tr>
<td>-p, --password.file.location</td>
<td>Specify the password file location</td>
</tr>
<tr>
<td>-t, --target</td>
<td>Specify the target application</td>
</tr>
<tr>
<td>-ts, --truststore.location</td>
<td>Specify the truststore location</td>
</tr>
</tbody>
</table>

8.20. 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 master to a slave. 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 master.
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 master replicator, the slaves 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:

```sh
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`

```sh
shell> cd ~
shell> wget https://launchpad.net/mysql-sandbox/mysql-sandbox-3/mysql-sandbox-3-3.0.44.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 master should be offline [212], 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-3.0.0-254.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 slaves to be sure they are online. The slaves services will process all extracted entries.

### 8.20.1. Provisioning from RDS

The `tungsten_provision_thl` script is designed to run from a replication master connected to a standard MySQL instance. The standard commands will not work if you are using RDS as a master.

The simplest method is to add the `--extract-from` 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 master at the current binary log position.

```
shell> tungsten_provision_thl
   --extract-from=rds
   --tungsten-replicator-package=/home/tungsten/tungsten-replicator-3.0.0-254.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` and `--extract-from-host` arguments. The script will read the correct master position based on the slave replication position. After completion, resume extraction from the master 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-3.0.0-254.tar.gz
   --mysql-package=/home/tungsten/mysql-5.6.20-linux-glibc2.5-x86_64.tar.gz
   --schemas=test
```

8.20.2. `tungsten_provision_thl` Reference

The format of the command is:

```
tungsten_provision_thl [ --cleanup-on-failure ] [ --clear-logs ] [ --directory ] [ --extract-from-rds-native-slaverdsrds-read-replicatungsten-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 [306]`

<table>
<thead>
<tr>
<th>Option</th>
<th>--cleanup-on-failure [306]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Cleanup the sandbox installations when the provision process fails</td>
</tr>
<tr>
<td>Value Type</td>
<td>boolean</td>
</tr>
<tr>
<td>Default</td>
<td>false</td>
</tr>
</tbody>
</table>

- `--clear-logs [306]`

<table>
<thead>
<tr>
<th>Option</th>
<th>--clear-logs [306]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Delete all THL and relay logs for the service</td>
</tr>
</tbody>
</table>

306
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>--directory</td>
<td>Use this installed Tungsten directory as the base for all operations</td>
<td>boolean</td>
</tr>
<tr>
<td>--extract-from</td>
<td>The type of server you are going to extract from</td>
<td>string</td>
</tr>
<tr>
<td>--extract-from-host</td>
<td>The hostname of a different MySQL server that will be used as the source for mysqldump</td>
<td>string</td>
</tr>
<tr>
<td>--extract-from-port</td>
<td>The listening port of a different MySQL server that will be used as the source for mysqldump</td>
<td>numeric</td>
</tr>
<tr>
<td>--help</td>
<td>Display the help message</td>
<td>string</td>
</tr>
<tr>
<td>--info</td>
<td>Provide information-level messages</td>
<td>string</td>
</tr>
<tr>
<td>--java-file-encoding</td>
<td>Java platform charset</td>
<td>string</td>
</tr>
<tr>
<td>--json</td>
<td></td>
<td>string</td>
</tr>
</tbody>
</table>
### Command-line Tools

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
<th>Default</th>
<th>Aliases</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--json</code></td>
<td>Provide return code and logging messages as a JSON object after the script finishes</td>
<td>string</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>--mysql-package</code></td>
<td>The location of the MySQL tar.gz package</td>
<td>string</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>--net-ssh-option</code></td>
<td>Sets additional options for SSH usage by the system, such as port numbers and passwords.</td>
<td>string</td>
<td>default</td>
<td></td>
</tr>
<tr>
<td><code>--notice</code></td>
<td>Provide notice-level messages</td>
<td>string</td>
<td></td>
<td><code>-n</code></td>
</tr>
<tr>
<td><code>--offline</code></td>
<td>Put required replication services offline before processing</td>
<td>boolean</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td><code>--online</code></td>
<td>Put required replication services online after successful processing</td>
<td>boolean</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td><code>--quiet</code></td>
<td>Execute with the minimum of output</td>
<td>string</td>
<td></td>
<td><code>-q</code></td>
</tr>
<tr>
<td><code>--sandbox-directory</code></td>
<td>The location to use for storing the temporary replicator and MySQL server</td>
<td>string</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For more information on the options, see [http://net-ssh.github.com/ssh/v2/api/classes/Net/SSH.html#M000002](http://net-ssh.github.com/ssh/v2/api/classes/Net/SSH.html#M000002).
### Command-line Tools

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>--sandbox-mysql-port</td>
<td>The listening port for the MySQL Sandbox</td>
<td>string</td>
<td>3307</td>
</tr>
<tr>
<td>--sandbox-password</td>
<td>The password for the MySQL sandbox user</td>
<td>string</td>
<td>secret</td>
</tr>
<tr>
<td>--sandbox-rmi-port</td>
<td>The listening port for the temporary Tungsten Replicator</td>
<td>string</td>
<td>10002</td>
</tr>
<tr>
<td>--sandbox-user</td>
<td>The MySQL user to create and use in the MySQL Sandbox</td>
<td>string</td>
<td>tungsten</td>
</tr>
<tr>
<td>--schemas</td>
<td>The provision process will be limited to these schemas</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>--service</td>
<td>Replication service to read information from</td>
<td>string</td>
<td>alpha</td>
</tr>
<tr>
<td>--tungsten-replicator-package</td>
<td>The location of a fresh Tungsten Replicator tar.gz package</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>--validate</td>
<td>Run the script validation for the provided options and files</td>
<td>boolean</td>
<td>false</td>
</tr>
<tr>
<td>--verbose</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
8.21. The `tungsten_provision_slave` Script

The script was added in Tungsten Replicator 2.2.0. It cannot be backported to older versions.

The `tungsten_provision_slave` script allows you to easily provision, or reprovision, a database server using information from a remote host. It implements the Tungsten Script Interface as well as these additional options.

```
tungsten_provision_slave [ --clear-logs ] [ --direct ] [ --directory ] [ -f, --force ] [ --help, -h ] [ --info, -i ] [ --json ] [ --mysqldump ] [ --net-ssh-option ] [ --notice, -n ] [ --offline ] [ --offline-timeout ] [ --online ] [ --service ] [ --source ] [ --source-directory ] [ --validate ] [ --verbose, -v ] [ --xtrabackup ]
```

Where:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--clear-logs</td>
<td>Delete all THL and relay logs for the service</td>
</tr>
<tr>
<td>--direct</td>
<td>Use the MySQL data directory for staging and preparation</td>
</tr>
<tr>
<td>--directory</td>
<td>The $CONTINUENT_ROOT directory to use for running this com-</td>
</tr>
<tr>
<td></td>
<td>mand. It will default to the directory you use to run the script.</td>
</tr>
<tr>
<td>-f, --force</td>
<td>Continue operation even if script validation fails</td>
</tr>
<tr>
<td>--help, -h</td>
<td>Show help text</td>
</tr>
<tr>
<td>--info, -i</td>
<td>Display info, notice, warning, and error messages</td>
</tr>
<tr>
<td>--json</td>
<td>Output all messages and the return code as a JSON object</td>
</tr>
<tr>
<td>--mysqldump</td>
<td>Use mysqldump for generating the information</td>
</tr>
<tr>
<td>--net-ssh-option</td>
<td>Display custom SSH options to use for SSH communication to other hosts.</td>
</tr>
<tr>
<td>--notice, -n</td>
<td>Put required replication services offline before processing</td>
</tr>
<tr>
<td>--offline</td>
<td>Put required replication services offline after successful processing</td>
</tr>
<tr>
<td>--source</td>
<td>Server to use as a source for the backup</td>
</tr>
<tr>
<td>--source-directory</td>
<td>Directory on --source to find installed software</td>
</tr>
<tr>
<td>--validate</td>
<td>Only run script validation</td>
</tr>
<tr>
<td>--verbose, -v</td>
<td>Show verbose information during processing</td>
</tr>
<tr>
<td>--xtrabackup</td>
<td>Use xtrabackup for generating the information</td>
</tr>
</tbody>
</table>

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 slave that is stopped, or if the slave is not in either the `ONLINE` or `OFFLINE:NORMAL` states. This can be overridden by using the `--force` option.

When provisioning masters, for example in fan-in, or when recovering a failed master in a standard master-slave topology, the service must be reset with the `trepctl reset` after the command is finished. The service must also be reset on all slaves.

The `--service` argument is used to determine which database server should be provisioned. If there are multiple services defined in the replicator and one of those is a master, the master service must be specified.

If the installation directory on `--source` is different from the target, specify `--source-directory` to specify where it can be found. This option should point to an installation that is running the `--service replication service`. The `--source-directory` option is not required if the software is installed to the same directory on both servers.
Using xtrabackup

The script will use Xtrabackup by default. It 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. Passing the `--direct` option will empty the MySQL data directory prior to doing the backup and place the streaming backup there. After taking the backup, the script will prepare the directory and restart the MySQL server.

Using mysqldump

If you have a small dataset or don’t have Xtrabackup, you may pass the `--mysqldump` option to use it. It implements the Tungsten Script Interface as well as these additional options.

Compatibility

The script only works with MySQL at this time.

8.22. The `tungsten_read_master_events` Script

The script was added in Tungsten Replicator 2.2.0. It cannot be backported to older versions.

The `tungsten_read_master_events` displays the raw contents of the master 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 ] [ --info, -i ] [ --json ] [ --low ] [ --net-ssh-option ] [ --notice, -n ] [ --service ] [ --source ] [ --validate ] [ --verbose, -v ]
```

Where:

Table 8.36. `tungsten_read_master_events` Command-line Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--directory</code></td>
<td>The CONTINUENT_ROOT directory to use for running this command. It will default to the directory you use to run the script.</td>
</tr>
<tr>
<td><code>--force</code></td>
<td>Continue operation even if script validation fails</td>
</tr>
<tr>
<td><code>--help, -h</code></td>
<td>Show help text</td>
</tr>
<tr>
<td><code>--high</code></td>
<td>Display events ending with this sequence number</td>
</tr>
<tr>
<td><code>--info, -i</code></td>
<td>Display info, notice, warning, and error messages</td>
</tr>
<tr>
<td><code>--json</code></td>
<td>Output all messages and the return code as a JSON object</td>
</tr>
<tr>
<td><code>--low</code></td>
<td>Display events starting with this sequence number</td>
</tr>
<tr>
<td><code>--net-ssh-option</code></td>
<td>Provide custom SSH options to use for SSH communication to other hosts.</td>
</tr>
<tr>
<td><code>--notice, -n</code></td>
<td>Display notice, warning, and error messages</td>
</tr>
<tr>
<td><code>--service</code></td>
<td>Replication service to read information from</td>
</tr>
<tr>
<td><code>--source</code></td>
<td>Determine metadata for the <code>--after</code>, <code>--low</code>, <code>--high</code> statements from this host</td>
</tr>
<tr>
<td><code>--validate</code></td>
<td>Only run script validation</td>
</tr>
<tr>
<td><code>--verbose, -v</code></td>
<td>Show verbose information during processing</td>
</tr>
</tbody>
</table>

Display all information after a specific sequence number

This may be used when you have had a master 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 master 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
```
8.23. The *tungsten_send_diag* Script

The script was added in Continuent Tungsten 5.2.0.

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 9.5.2, "tpm diag Command" to generate a diagnostic package.

```
tungsten_send_diag [ --case, -c ][ --contentType ][ --debug ][ --diag, -d ][ --email, -e ][ --file, -f ][ --help, -h ][ --tpm, -t ][ --verbose, -v ]
```

Where:

Table 8.37. *tungsten_send_diag* Command-line Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--case, -c</td>
<td>Specify the support case number</td>
</tr>
<tr>
<td>--contentType</td>
<td>Specify the Content-Type for a file you are uploading</td>
</tr>
<tr>
<td>--debug</td>
<td>Debug mode is VERY chatty, avoid it unless you really need it.</td>
</tr>
<tr>
<td>--diag, -d</td>
<td>Automatically generate a tpm diag zip file and upload it</td>
</tr>
<tr>
<td>--email, -e</td>
<td>Email address to embed into the uploaded file name</td>
</tr>
<tr>
<td>--file, -f</td>
<td>File name to upload</td>
</tr>
<tr>
<td>--help, -h</td>
<td>Show help text</td>
</tr>
<tr>
<td>--tpm, -t</td>
<td>Full path to the tpm command you wish to use to execute a tpm diag</td>
</tr>
<tr>
<td>--verbose, -v</td>
<td>Show verbose output</td>
</tr>
</tbody>
</table>

You must specify either `--diag`, `--tpm`, or `--file`, but not both. For example:

```
shell> tungsten_send_diag --diag -c 1234
```

You must specify either `--email` or `--case`, and you may provide both if you wish. For example:

```
shell> tungsten_send_diag --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 --tpm -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 --tpm -c 1234 -t /opt/continuent/tungsten/tools/tpm -t /opt/replicator/tungsten/tools/tpm
```

8.24. The *tungsten_set_position* Script

The script was added in Tungsten Replicator 2.2.0. It cannot be backported to older versions.

The *tungsten_set_position* updates the `trep_commit_seqno` table to reflect the given THL sequence number or provided information. It implements the Tungsten Script Interface as well as these additional options.

```
tungsten_set_position [ --clear-logs ][ --epoch ][ --event-id ][ --high ][ --low ][ --offline ][ --offline-timeout ][ --online ][ --replicate-statements ][ --seqlno ][ --service ][ --source ][ --source-directory ][ --source-id ][ --sql ]
```

Where:

Table 8.38. *tungsten_set_position* Command-line Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--clear-logs</td>
<td>Delete all THL and relay logs for the service</td>
</tr>
</tbody>
</table>
## Command-line Tools

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--epoch</td>
<td>The epoch number to use for updating the trep_commit_seqno table</td>
</tr>
<tr>
<td>--event-id</td>
<td>The event id to use for updating the trep_commit_seqno table</td>
</tr>
<tr>
<td>--high</td>
<td>Display events ending with this sequence number</td>
</tr>
<tr>
<td>--low</td>
<td>Display events starting with this sequence number</td>
</tr>
<tr>
<td>--offline</td>
<td>Put required replication services offline before processing</td>
</tr>
<tr>
<td>--offline-timeout</td>
<td>Put required replication services offline before processing</td>
</tr>
<tr>
<td>--online</td>
<td>Put required replication services online after successful processing</td>
</tr>
<tr>
<td>--replicate-statements</td>
<td>Execute the events so they will be replicated if the service is a master</td>
</tr>
<tr>
<td>--seqno</td>
<td>The sequence number to use for updating the trep_commit_seqno table</td>
</tr>
<tr>
<td>--service</td>
<td>Replication service to read information from</td>
</tr>
<tr>
<td>--source</td>
<td>Determine metadata for the --after, --low, --high statements from this host</td>
</tr>
<tr>
<td>--source-directory</td>
<td>Directory on --source to find installed software</td>
</tr>
<tr>
<td>--source-id</td>
<td>The source id to use for updating the trep_commit_seqno table</td>
</tr>
<tr>
<td>--sql</td>
<td>Only output the SQL statements needed to update the schema</td>
</tr>
</tbody>
</table>

### General Operation

In order to update the trep_commit_seqno table, the replication service must be offline. You may pass the `--offline` option to do that for you. The `--online` option will put the replication services back online at successful completion.

In most cases you will want to pass the `--clear-logs` argument so that all THL and relay logs are delete from the server following provisioning. This ensures that any corrupted or inconsistent THL records are removed prior to replication coming back online.

The `--service` argument is used to determine which database server should be provisioned.

If the installation directory on `--source` is different from the target, specify `--source-directory` to specify where it can be found. This option should point to an installation that is running the `--service` replication service. The `--source-directory` option is not required if the software is installed to the same directory on both servers.

This command will fail if there is more than one record in the trep_commit_seqno table. This may happen if parallel replication does not stop cleanly. You may bypass that error with the `--force` option.

### Update trep_commit_seqno with information from a THL event

This will read the THL information from the host specified as `--source`.

```
shell> tungsten_set_position --seqno=5273 --source=db1
```

### Update trep_commit_seqno with specific information

The script will also accept specific values to update the trep_commit_seqno table. This may be used when bringing a new master service online or when the THL event is no longer available.

```
shell> tungsten_set_position --seqno=5273 --epoch=5264
   --source-id=db1

shell> tungsten_set_position --seqno=5273 --epoch=5264
   --source-id=db1 --event-id=mysql-bin.000025:0000000000000421
```

### Compatibility

The script only works with MySQL at this time.

### 8.25. The undeployall Command

The `undeployall` command removes startup the startup and reboot scripts created 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:
Removing any system startup links for /etc/init.d/treplicator ...

To enable the scripts on the system, use `deployall`.

### 8.26. The `updateCDC.sh` Command

The `updateCDC.sh` script updates and existing configuration for Oracle CDC, updating for new tables and user/password configuration.

The script accepts one argument, the filename of the configuration file that will define the CDC configuration. The file accepts the parameters as listed in Table 8.19, "setupCDC.conf Configuration Options".

To use, supply the name of the configuration file:

```bash
sh> ./updateCDC.sh sample.conf
```
Chapter 9. The tpm Deployment Command

tpm, or the Tungsten Package Manager, is a complete configuration, installation and deployment tool for Tungsten Clustering. 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 Clustering 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 created on each host that will run Tungsten Clustering. 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 Clustering.

• 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 Tungsten Clustering. 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 Clustering 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 Tungsten Clustering 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 with the Tungsten Clustering 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 Clustering 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 Tungsten Clustering 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.
### Table 9.1. TPM Deployment Methods

<table>
<thead>
<tr>
<th>Feature</th>
<th>Staging Directory</th>
<th>INI File</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploy Multiple Services</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Deploy to Multiple Hosts</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Individual Host-based Configuration</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Single-Step Upgrade</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Requires SSH Configuration</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>RPM/PKG Support</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**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-5.4.1-41`. 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 Clustering 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 Clustering.
   b. Complete all the Appendix B, Prerequisites.
The tpm Deployment Command

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 Tungsten Clustering and deployment files to each server
  
  During this stage part of the Tungsten Clustering package is copied to each server. At this point 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.

```plaintext
# 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 Tungsten Clustering and write configuration files

  If validation is successful, we will move on to deploying Tungsten Clustering and writing the actual configuration files. The tpm command uses a JSON file that summarizes the configuration. The Tungsten Clustering 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 of Tungsten Clustering 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 Tungsten Clustering 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 Clustering services
After Tungsten Clustering is fully configured, the `tpm` command will start services on all of the hosts. This process is slightly different depending on if you are doing a clean install or an upgrade.

- **Install**
  1. Check if `--start` or `--start-and-report` were provided in the configuration
  2. Start the Tungsten Replicator and Tungsten Manager on all hosts
  3. Wait for the Tungsten Manager to become responsive
  4. Start the Tungsten Connector 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 the 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.

```
shells> 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 9.4, “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

```
shells> ./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

```
shells> ./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 [386], --slaves [398] and --master [386]**

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 [386]` and `--slaves [398]; --members [386]` will be calculated as the unique join of both values.

If you specify `--master [386]` and `--members [386]; --slaves [398]` will be calculated as the unique difference of both values.
9.3.3. Configuring a single host

This will apply the --repl-backup-method option to just the host3 server. Multiple hosts may be given as a comma-separated list. The names used in the --members, --slaves, --master options should be used when calling --hosts. 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.

9.3.5. Installation

After you have prepared the configuration file, it is time to 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, --slaves, and --master.

9.3.5.1. Installing a set of specific services

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

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.

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.

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
```

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/continuent-tungsten-2.0.3-519
```

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-5.4.1-41.tar.gz

2. Create the text file config.sh based on the output from tpm reverse:
   
   shell> cd tungsten-replicator-5.4.1-41
   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-5.4.1-41
   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

   In Tungsten Replicator 3.0 and later, tpm will automatically search all tungsten*.ini files within the /etc/tungsten directory. An alternative directory can be searched using --ini [334] option to tpm.

   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
### 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.

```sh
shell> cd /opt/continuent/software/tungsten-replicator-5.4.1-41
shell> /tools/tpm install
```

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.

```sh
shell> cd /opt/continuent/software
shell> tar zxf tungsten-replicator-5.4.1-41.tar.gz
shell> cd tungsten-replicator-5.4.1-41
shell> /tools/tpm update
```

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 master 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 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 master 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`.

```sh
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.

```sh
shell> sudo mkdir /etc/tungsten
shell> sudo chown -R tungsten: /etc/tungsten
shell> chmod 700 /etc/tungsten
shell> touch /etc/tungsten/tungsten.ini
```
The **tpm Deployment Command**

```shell
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
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
shell> vim /etc/tungsten/tungsten.ini
:%s/tools\^tpm configure /[g
:%s/^--//g
:%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.

1. **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
shell> cd /opt/continuent/software
shell> tar zxf tungsten-replicator-5.4.1-41.tar.gz
```

2. **On each node, change to the extracted directory and execute the `tpm` command**:

```shell
shell> cd /opt/continuent/software/tungsten-replicator-5.4.1-41
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
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
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
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.
The tpm Deployment Command

```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**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--force [325], -f [325]</td>
<td>Do not display confirmation prompts or stop the configure process for errors</td>
</tr>
<tr>
<td>-help [325], -h [325]</td>
<td>Displays help message</td>
</tr>
<tr>
<td>--info [325], -i [325]</td>
<td>Display info, notice, warning and error messages</td>
</tr>
<tr>
<td>--notice [325], -n [325]</td>
<td>Display notice, warning and error messages</td>
</tr>
<tr>
<td>--preview [325], -p [325]</td>
<td>Displays the help message and preview the effect of the command line options</td>
</tr>
<tr>
<td>--profile file [325]</td>
<td>Sets name of config file</td>
</tr>
<tr>
<td>--quiet [325], -q [325]</td>
<td>Only display warning and error messages</td>
</tr>
<tr>
<td>--verbose [325], -v [325]</td>
<td>Display debug, info, notice, warning and error messages</td>
</tr>
</tbody>
</table>

- **--force [325]**
  Forces the deployment process to complete even if there are warning or error messages that would normally cause the process to fail. Forcing the installation also ignores all confirmation prompts during installation and always attempts to complete the process.

- **--help [325]**
  Displays the help message for tpm showing the current options, commands and version information.

- **--info [325]**
  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 [325].

- **--notice [325]**
  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 [325]**

- **--profile file [325]**
  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 [325]**
  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 [325]**
  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 as well as complex multi-master replication services. The coordination requires SSH connections between the hosts according to the Appendix B, Prerequisites. There are two exceptions for this:

1. When the --hosts [382] 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 [386], --slaves [398], --master [386] arguments should be used when calling --hosts [382].

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 Clustering 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

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>configure</td>
<td>Configure a data service within the global configuration</td>
</tr>
<tr>
<td>diag</td>
<td>Obtain diagnostic information</td>
</tr>
<tr>
<td>fetch</td>
<td>Fetch configuration information from a running service</td>
</tr>
<tr>
<td>firewall</td>
<td>Display firewall information for the configured services</td>
</tr>
<tr>
<td>help</td>
<td>Show command help information</td>
</tr>
<tr>
<td>install</td>
<td>Install a data service based on the existing and runtime parameters</td>
</tr>
<tr>
<td>mysql</td>
<td>Open a connection to the configured MySQL server</td>
</tr>
<tr>
<td>query</td>
<td>Query the active configuration for information</td>
</tr>
<tr>
<td>reset</td>
<td>Reset the cluster on each host</td>
</tr>
<tr>
<td>reset-thl</td>
<td>Reset the THL for a host</td>
</tr>
<tr>
<td>ssh-copy-cert</td>
<td>Executes the commands required to generate authorized SSH certificate and keys required for tpm</td>
</tr>
<tr>
<td>uninstall</td>
<td>Uninstall software from host(s)</td>
</tr>
<tr>
<td>update</td>
<td>Update an existing configuration or software version</td>
</tr>
<tr>
<td>validate</td>
<td>Validate the current configuration</td>
</tr>
<tr>
<td>validate-update</td>
<td>Validate the current configuration and update</td>
</tr>
</tbody>
</table>

#### 9.5.1. tpm configure Command

The configure command to tpm creates a configuration file within the current profiles directory.

#### 9.5.2. tpm diag Command

The tpm diag command will create a ZIP file including log files and current dataservice status. It will connect to all servers listed in the tpm reverse output attempting to collect information.

```shell
tpm diag
```

NOTE: >> host1 >> Diagnostic information written to /home/tungsten/tungsten-dia-2013-10-09-21-04-23.zip

The information collected depends on the installation type:

- Within a staging directory installation, all the hosts configured within the cluster will be contacted, and all the information across all hosts will be incorporated into the Zip file that is created.
- Within an INI installation, the other hosts in the cluster will be contacted if ssh has been configured and the other hosts can be reached. If ssh is not available, a warning will be printed, and each host will need to be accessed individually to run tpm diag.

5.3.7 and 6.0.5 Behavior Change. As of versions 5.3.7 and 6.0.5, the tpm diag command behavior has been modified.

- With Staging-method deployments, the tpm diag command continues to behave as before:
  - The tpm diag command alone will obtain diagnostics from all hosts in the cluster.
- With INI-method deployments, the new behavior is as follows:
  - The tpm diag command alone will obtain diagnostics from the local host only.
  - The tpm diag command along with the specified host(s) will obtain diagnostics from the specified host(s) only.

**Warning**

Limitation: the host list MUST include the local hostname or the command will fail.
9.5.3. 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
dir/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` 382
  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` 403
  The username to be used when logging in to other hosts.

- `--directory` 378
  The installation directory of the current Tungsten Clustering installation. If `autodetect` is specified, then `tpm` will look for the installation directory by checking any running Tungsten Clustering processes.

9.5.4. tpm firewall Command

The `tpm firewall` command displays port information required to configure 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.

9.5.5. 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
```
9.5.6. 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 Clustering 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.

9.5.7. tpm mysql Command

This will open a MySQL CLI connection to the local MySQL server using the current values for `--replication-user [397]`, `--replication-password [396]` and `--replication-port [396].`

```
shell> /tools/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.

9.5.8. 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 Clustering was installed
• **tpm query values** — return the list of configured values
• **tpm query version** — get the version of the current installation

### 9.5.8.1 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:

```python
shell> tpm query config
{
  "__system_defaults_will_be_overwritten__": {
    ...
    "staging_directory": "/home/tungsten/tungsten-replicator-5.4.1-41",
    "staging_host": "tr-ms1",
    "staging_user": "tungsten"
  }
}
```

### 9.5.8.2 tpm query dataservices

Returns the list of configured dataservices that have, or will be, installed:

```python
shell> tpm query dataservices
alpha : PHYSICAL
```

### 9.5.8.3 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:

```python
shell> tpm query deployments
{
  "config_target_basename": "tungsten-replicator-5.4.1-41_pid22729",
  "dataservice_host_options": {
    'alpha': {
      'start': "true"
    }
  }
  ...
  "staging_directory": "/home/tungsten/tungsten-replicator-5.4.1-41",
  "staging_host": "tr-ms1",
  "staging_user": "tungsten"
}
```

### 9.5.8.4 tpm query manifest

Returns the manifest information for the identified release of Tungsten Clustering, including the build, source and component versions, returned in the form of a JSON value:

```python
shell> tpm query manifest
{
  "SVN": {
    "bristlecone": {
      "URL": "http://bristlecone.googlecode.com/svn/trunk/bristlecone",
      "revision": 178
    },
    "commons": {
      "URL": "https://tungsten-replicator.googlecode.com/svn/trunk/commons",
      "revision": 1983
    },
    "cookbook": {
      "URL": "https://tungsten-toolbox.googlecode.com/svn/trunk/cookbook",
      "revision": 230
    },
    "replicator": {
      "URL": "https://tungsten-replicator.googlecode.com/svn/trunk/replicator",
      "revision": 1983
    }
  },
  "date": "Wed Jan  8 18:11:08 UTC 2014",
  "host": "ip-10-250-35-36",
}
```
9.5.8.5. 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.

9.5.8.6. 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-5.4.1-41
```

This can be useful when the installation host and directory from which the original configuration was made need to be updated or modified.

9.5.8.7. tpm query version

Returns the version for the identified version of Tungsten Clustering:

```
shell> tpm query version
  5.4.1-41
```

9.5.9. 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 [382]` option.

```
shell> ./tools/tpm reset
```

9.5.10. 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 [382]` option.

```
shell> ./tools/tpm reset-thl
```
The tpm Deployment Command

9.5.11. 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> ./tools/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
  --master=host1
  --members=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

9.5.12. tpm ssh-copy-cert Command

The tpm ssh-copy-cert command executes all the required commands to generate the required ssh certificates required for SSH operation by tpm. Executing the command should generate the required directory, certificate and add that information to the required SSH files, then ensure that the directory permissions and ownership on ~/.ssh are set correctly.

For example, executing the command outputs the stages and progress:

```
shell> ./tools/tpm ssh-copy-cert
mkdir -p ~/.ssh
echo "-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAnMSRTwBB2Ik6FOTZYxQkXglFivniLSRxcNw73UDVEGps7D0
Peo2XH4tX1sHPXPAcJib6j6Sn0ZjRjWk4oHhXs2VsStSsvX61vVVRkN7oX3Yw
L8Kz2jT7hj6hBbukLkU9F0mR0Zo49G29JYj34l6awork6U2LIlufgp3L2yQfz3b
Oq8Kz2jT7hj6hBbukLkU9F0mR0Zo49G29JYj34l6awork6U2LIlufgp3L2yQfz3b
-----END RSA PRIVATE KEY-----" > ~/.ssh/id_rsa
echo "ssh-rsa ...
```

9.5.13. 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> ./tools/tpm uninstall --i-am-sure
```

### 9.5.14. 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/tungsten-replicator-5.4.1-41`. 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

As of software version 4.0.0, 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.

### 9.5.15. 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> ./tools/tpm validate
```

Validation failed
9.5.16. 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> ./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 [335]
- In an INI file, without the double-dash prefix, i.e. skip-validation-check=MySQLConnectorPermissionsCheck [335]

Table 9.4. tpm Common Options

<table>
<thead>
<tr>
<th>CmdLine Option</th>
<th>INI File Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--enable-validation-check [333]</td>
<td>enable-validation-check [333]</td>
<td>Enable a specific validation check, overriding any configured skipped checks</td>
</tr>
<tr>
<td>--enable-validation-warnings [334]</td>
<td>enable-validation-warnings [334]</td>
<td>Enable a specific validation warning, overriding any configured skipped warning</td>
</tr>
<tr>
<td>--ini [334]</td>
<td>ini [334]</td>
<td>Specify the location of the directory where INI files will be located</td>
</tr>
<tr>
<td>--net-ssh-option [334]</td>
<td>net-ssh-option [334]</td>
<td>Set the Net:SSH option for remote system calls</td>
</tr>
<tr>
<td>--property [334], --property=tykey=value [334], --property=tykey=value [334], --property=tykey=match/replace [334]</td>
<td>property [334], property=key=value [334], property=key=value [334], property=key=match/replace [334]</td>
<td>Modify specific property values for the key in any file that the configuration script touches.</td>
</tr>
<tr>
<td>--remove-property [335]</td>
<td>remove-property [335]</td>
<td>Remove the setting for a previously configured property</td>
</tr>
</tbody>
</table>

--enable-validation-check [333]
The **tpm Deployment Command**

<table>
<thead>
<tr>
<th>Value Type</th>
<th>string</th>
</tr>
</thead>
</table>

The **--enable-validation-check** 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** and **--enable-validation-check** is equivalent to explicitly disabling the specified check.

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>--enable-validation-warnings</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

The **--enable-validation-warnings** 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`. If a check fails, installation is canceled.

Setting both **--skip-validation-warnings** and **--enable-validation-warnings** is equivalent to explicitly disabling the specified check.

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>--ini</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

Specifies an alternative location for the INI files from the default.

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>--net-ssh-option</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

Enables you to set a specific Net::SSH option. For example:

```
shell> tpm update ... --net-ssh-option=compression=zlib
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>--property</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

The **--property** 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`.
The tpm Deployment Command

---remove-property [335]

<table>
<thead>
<tr>
<th>Option</th>
<th>--remove-property [335]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>remove-property [335]</td>
</tr>
<tr>
<td>Description</td>
<td>Remove the setting for a previously configured property</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

Remove a previous explicit property setting. For example:

```
shell> tpm configure --remove-property=replicator.filter.pkey.addPkeyToInserts
```

---skip-validation-check [335]

<table>
<thead>
<tr>
<th>Option</th>
<th>--skip-validation-check [335]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>skip-validation-check [335]</td>
</tr>
<tr>
<td>Description</td>
<td>Do not run the specified validation check.</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

The --skip-validation-check [335] 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 [346], and could be ignored using --skip-validation-check=MySQLDefaultTableTypeCheck [335].

Setting both --skip-validation-check [335] and --enable-validation-check [333] is equivalent to explicitly disabling the specified check.

---skip-validation-warnings [335]

<table>
<thead>
<tr>
<th>Option</th>
<th>--skip-validation-warnings [335]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>skip-validation-warnings [335]</td>
</tr>
<tr>
<td>Description</td>
<td>Do not display warnings for the specified validation check.</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

The --skip-validation-warnings [335] 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 [346], and could be ignored using --skip-validation-warnings=SwappinessCheck [335].

Setting both --skip-validation-warnings [335] and --enable-validation-warnings [334] 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 [335] or --skip-validation-check [335] options.
## Table 9.5. *tpm* Validation Checks

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BackupDirectoryWriteableCheck [339]</td>
<td>Checks that the configured backup directory is writeable</td>
</tr>
<tr>
<td>BackupDumpDirectoryWriteableCheck [339]</td>
<td>Checks the backup temp directory is writeable</td>
</tr>
<tr>
<td>BackupScriptAvailableCheck [339]</td>
<td>Checks that the configured backup script exists and can be executed</td>
</tr>
<tr>
<td>ClusterDiagnosticsCheck [339]</td>
<td></td>
</tr>
<tr>
<td>ClusterStatusCheck [339]</td>
<td></td>
</tr>
<tr>
<td>ConfigurationStorageDirectoryCheck [340]</td>
<td></td>
</tr>
<tr>
<td>ConfigureValidationCheck [340]</td>
<td></td>
</tr>
<tr>
<td>ConfiguredDirectoryCheck [340]</td>
<td></td>
</tr>
<tr>
<td>ConflictingReplicationServiceTHLPortsCheck [340]</td>
<td></td>
</tr>
<tr>
<td>ConnectorChecks [340]</td>
<td>Ensures that the configured connector selection is valid</td>
</tr>
<tr>
<td>ConnectorDBVersionCheck [340]</td>
<td></td>
</tr>
<tr>
<td>ConnectorListenerAddressCheck [340]</td>
<td></td>
</tr>
<tr>
<td>ConnectorRWROAddressesCheck [340]</td>
<td>Ensure the RW and RO addresses are different</td>
</tr>
<tr>
<td>ConnectorSmartScaleAllowedCheck [340]</td>
<td>Confirms whether SmartScale is valid within the current configured parameters</td>
</tr>
<tr>
<td>ConnectorUserCheck [340]</td>
<td></td>
</tr>
<tr>
<td>ConsistentReplicationCredentialsCheck [341]</td>
<td></td>
</tr>
<tr>
<td>CurrentCommandCoordinatorCheck [341]</td>
<td></td>
</tr>
<tr>
<td>CurrentConnectorCheck [341]</td>
<td></td>
</tr>
<tr>
<td>CurrentReleaseDirectoryIsSymlink [341]</td>
<td></td>
</tr>
<tr>
<td>CurrentTopologyCheck [341]</td>
<td></td>
</tr>
<tr>
<td>CurrentVersionCheck [341]</td>
<td></td>
</tr>
<tr>
<td>DatasourceBootScriptCheck [341]</td>
<td></td>
</tr>
<tr>
<td>DifferentMasterSlaveCheck [341]</td>
<td></td>
</tr>
<tr>
<td>DirectOracleServiceSIDCheck [341]</td>
<td></td>
</tr>
<tr>
<td>ElasticsearchValidationCheck [341]</td>
<td></td>
</tr>
<tr>
<td>EncryptionCheck [341]</td>
<td></td>
</tr>
<tr>
<td>EncryptionKeystoreCheck [342]</td>
<td></td>
</tr>
<tr>
<td>FileValidationCheck [342]</td>
<td></td>
</tr>
<tr>
<td>FirewallCheck [342]</td>
<td></td>
</tr>
<tr>
<td>GlobalHostAddressesCheck [342]</td>
<td></td>
</tr>
<tr>
<td>GlobalHostOracleLibrariesFoundCheck [342]</td>
<td></td>
</tr>
<tr>
<td>GlobalMatchingPingMethodCheck [342]</td>
<td></td>
</tr>
<tr>
<td>GlobalRestartComponentsCheck [342]</td>
<td></td>
</tr>
<tr>
<td>GroupValidationCheck [342]</td>
<td></td>
</tr>
<tr>
<td>HdfsValidationCheck [342]</td>
<td></td>
</tr>
<tr>
<td>HostLicensesCheck [342]</td>
<td></td>
</tr>
<tr>
<td>HostOracleLibrariesFoundCheck [342]</td>
<td></td>
</tr>
<tr>
<td>HostReplicatorServiceRunningCheck [343]</td>
<td></td>
</tr>
<tr>
<td>HostSkippedChecks [343]</td>
<td></td>
</tr>
<tr>
<td>HostnameCheck [343]</td>
<td></td>
</tr>
<tr>
<td>HostsFileCheck [343]</td>
<td></td>
</tr>
<tr>
<td>InstallServicesCheck [343]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>InstallationScriptCheck [343]</td>
<td>Checks whether a master host has been defined for the configured service.</td>
</tr>
<tr>
<td>InstallerMasterSlaveCheck [343]</td>
<td></td>
</tr>
<tr>
<td>InstallingOverExistingInstallation [343]</td>
<td></td>
</tr>
<tr>
<td>JavaUserTimeZoneCheck [343]</td>
<td></td>
</tr>
<tr>
<td>JavaVersionCheck [343]</td>
<td></td>
</tr>
<tr>
<td>KeystoresCheck [343]</td>
<td></td>
</tr>
<tr>
<td>KeystoresToCommitCheck [344]</td>
<td></td>
</tr>
<tr>
<td>ManagerActiveWitnessConversionCheck [344]</td>
<td></td>
</tr>
<tr>
<td>ManagerCheck [344]</td>
<td></td>
</tr>
<tr>
<td>ManagerHeapThresholdCheck [344]</td>
<td></td>
</tr>
<tr>
<td>ManagerListenerAddressCheck [344]</td>
<td></td>
</tr>
<tr>
<td>ManagerPingMethodCheck [344]</td>
<td></td>
</tr>
<tr>
<td>ManagerWitnessAvailableCheck [344]</td>
<td></td>
</tr>
<tr>
<td>ManagerWitnessNeededCheck [344]</td>
<td></td>
</tr>
<tr>
<td>MatchingHomeDirectoryCheck [344]</td>
<td></td>
</tr>
<tr>
<td>MissingReplicationServiceConfigurationCheck [344]</td>
<td></td>
</tr>
<tr>
<td>ModifiedConfigurationFilesCheck [344]</td>
<td></td>
</tr>
<tr>
<td>MySQLAllowIntensiveChecks [345]</td>
<td>Enables searching MySQL INFORMATION_SCHEMA for validation checks</td>
</tr>
<tr>
<td>MySQLApplierLogsCheck [345]</td>
<td></td>
</tr>
<tr>
<td>MySQLApplierPortCheck [345]</td>
<td></td>
</tr>
<tr>
<td>MySQLApplierServerIDCheck [345]</td>
<td></td>
</tr>
<tr>
<td>MySQLAvailableCheck [345]</td>
<td>Checks if MySQL is installed</td>
</tr>
<tr>
<td>MySQLBinaryLogsEnabledCheck [345]</td>
<td>Checks that binary logging has been enabled on MySQL</td>
</tr>
<tr>
<td>MySQLBinLogDoDBCheck [345]</td>
<td></td>
</tr>
<tr>
<td>MySQLClientCheck [345]</td>
<td>Checks whether the MySQL client command tool is available</td>
</tr>
<tr>
<td>MySQLConfigFileCheck [345]</td>
<td>Checks the existence of a MySQL configuration file</td>
</tr>
<tr>
<td>MySQLConnectorBridgeModePermissionsCheck [345]</td>
<td></td>
</tr>
<tr>
<td>MySQLConnectorPermissionsCheck [346]</td>
<td></td>
</tr>
<tr>
<td>MySQLDefaultTableTypeCheck [346]</td>
<td>Checks the default table type for MySQL</td>
</tr>
<tr>
<td>MySQLDumpCheck [346]</td>
<td>Checks that the mysqldump command version matches the installed MySQL</td>
</tr>
<tr>
<td>MySQLGeneratedColumnCheck [346]</td>
<td>Checks whether MySQL virtual/generated columns are defined</td>
</tr>
<tr>
<td>MySQLInnoDBEnabledCheck [346]</td>
<td></td>
</tr>
<tr>
<td>MySQLJsonDataTypeCheck [346]</td>
<td></td>
</tr>
<tr>
<td>MySQLLoadDataInfilePermissionsCheck [346]</td>
<td></td>
</tr>
<tr>
<td>MySQLLoginCheck [346]</td>
<td>Checks whether Tungsten Clustering can connect to MySQL using the configured credentials</td>
</tr>
<tr>
<td>MySQLMyISAMCheck [346]</td>
<td>Checks for the existence of MyISAM tables</td>
</tr>
<tr>
<td>MySQLMySQLReplicationCheck [347]</td>
<td></td>
</tr>
<tr>
<td>MySQLPasswordSettingCheck [347]</td>
<td></td>
</tr>
<tr>
<td>MySQLPermissionsCheck [347]</td>
<td></td>
</tr>
<tr>
<td>MySQLReadableLogsCheck [347]</td>
<td></td>
</tr>
<tr>
<td>MySQLSettingsCheck [347]</td>
<td></td>
</tr>
<tr>
<td>MySQLSuperReadOnlyCheck [347]</td>
<td>Checks whether super_read_only has been enabled on MySQL</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MySQLTriggerCheck [347]</td>
<td></td>
</tr>
<tr>
<td>MySQLUnsupportedDataTypesCheck [347]</td>
<td></td>
</tr>
<tr>
<td>MysqlConnectorCheck [348]</td>
<td></td>
</tr>
<tr>
<td>MysqldumpAvailableCheck [348]</td>
<td></td>
</tr>
<tr>
<td>MysqldumpSettingsCheck [348]</td>
<td></td>
</tr>
<tr>
<td>NmDirectoryRequiredCheck [348]</td>
<td></td>
</tr>
<tr>
<td>NtpdRunningCheck [348]</td>
<td></td>
</tr>
<tr>
<td>OSCheck [348]</td>
<td></td>
</tr>
<tr>
<td>OldServicesRunningCheck [348]</td>
<td></td>
</tr>
<tr>
<td>OpenFilesInitCheck [348]</td>
<td></td>
</tr>
<tr>
<td>OpensslLibraryCheck [348]</td>
<td></td>
</tr>
<tr>
<td>OracleLoginCheck [348]</td>
<td></td>
</tr>
<tr>
<td>OraclePermissionsCheck [348]</td>
<td></td>
</tr>
<tr>
<td>OracleRedoHeaderMinerDirectoryCheck [349]</td>
<td></td>
</tr>
<tr>
<td>OracleServiceSIDCheck [349]</td>
<td></td>
</tr>
<tr>
<td>OracleVersionCheck [349]</td>
<td></td>
</tr>
<tr>
<td>PGAvailableCheck [349]</td>
<td></td>
</tr>
<tr>
<td>ParallelReplicationCheck [349]</td>
<td></td>
</tr>
<tr>
<td>ParallelReplicationCountCheck [349]</td>
<td></td>
</tr>
<tr>
<td>PgControlAvailableCheck [349]</td>
<td></td>
</tr>
<tr>
<td>PgStandbyAvailableCheck [349]</td>
<td></td>
</tr>
<tr>
<td>PgdumpAvailableCheck [349]</td>
<td></td>
</tr>
<tr>
<td>PingSyntaxCheck [349]</td>
<td></td>
</tr>
<tr>
<td>PortAvailabilityCheck [350]</td>
<td></td>
</tr>
<tr>
<td>ProfileScriptCheck [350]</td>
<td></td>
</tr>
<tr>
<td>RMIListenerAddressCheck [350]</td>
<td></td>
</tr>
<tr>
<td>RelayDirectoryWriteableCheck [350]</td>
<td>Checks that the relay log directory can be written to</td>
</tr>
<tr>
<td>ReplicatorChecks [350]</td>
<td></td>
</tr>
<tr>
<td>RestartComponentsCheck [350]</td>
<td></td>
</tr>
<tr>
<td>RouterAffinityCheck [350]</td>
<td></td>
</tr>
<tr>
<td>RouterBridgeModeDefaultCheck [350]</td>
<td></td>
</tr>
<tr>
<td>RouterDelayBeforeOfflineCheck [350]</td>
<td></td>
</tr>
<tr>
<td>RouterKeepAliveTimeoutCheck [350]</td>
<td></td>
</tr>
<tr>
<td>RowBasedBinaryLoggingCheck [350]</td>
<td>Checks that Row-based binary logging has been enabled for heterogeneous deployments</td>
</tr>
<tr>
<td>RsyncAvailableCheck [351]</td>
<td></td>
</tr>
<tr>
<td>RubyVersionCheck [351]</td>
<td></td>
</tr>
<tr>
<td>SSHLoginCheck [351]</td>
<td>Checks connectivity to other hosts over SSH</td>
</tr>
<tr>
<td>ServiceTransferredLogStorageCheck [351]</td>
<td></td>
</tr>
<tr>
<td>StartingStoppedServices [351]</td>
<td></td>
</tr>
<tr>
<td>SudoCheck [351]</td>
<td></td>
</tr>
<tr>
<td>SwappinessCheck [351]</td>
<td>Checks the swappiness OS configuration is within a recommended range</td>
</tr>
<tr>
<td>THLDirectoryWriteableCheck [352]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>THLListenerAddressCheck [352]</td>
<td></td>
</tr>
<tr>
<td>THLSchemaChangeCheck [352]</td>
<td>Ensures that the existing THL format is compatible with the new release</td>
</tr>
<tr>
<td>THLStorageCheck [352]</td>
<td>Confirms the THL storage directory exists, is empty and writeable</td>
</tr>
<tr>
<td>THLStorageChecksum [352]</td>
<td></td>
</tr>
<tr>
<td>TargetDirectoryDoesNotExist [352]</td>
<td></td>
</tr>
<tr>
<td>TransferredLogStorageCheck [352]</td>
<td></td>
</tr>
<tr>
<td>UpgradeSameProductCheck [352]</td>
<td>Ensures that the same product is being updated</td>
</tr>
<tr>
<td>VIPEnabledHostAllowsRootCommands [352]</td>
<td></td>
</tr>
<tr>
<td>VIPEnabledHostArpPath [352]</td>
<td></td>
</tr>
<tr>
<td>VIPEnabledHostIfconfigPath [353]</td>
<td></td>
</tr>
<tr>
<td>VerticaUserGroupsCheck [353]</td>
<td>Checks that the Vertica user has the correct OS group membership</td>
</tr>
<tr>
<td>WhichAvailableCheck [353]</td>
<td>Checks the existence of a working which command</td>
</tr>
<tr>
<td>WriteableHomeDirectoryCheck [353]</td>
<td>Ensures the home directory can be written to</td>
</tr>
<tr>
<td>WriteableTempDirectoryCheck [353]</td>
<td>Ensures the temporary directory can be written to</td>
</tr>
<tr>
<td>XtrabackupAvailableCheck [353]</td>
<td></td>
</tr>
<tr>
<td>XtrabackupDirectoryWriteableCheck [353]</td>
<td></td>
</tr>
<tr>
<td>XtrabackupSettingsCheck [353]</td>
<td></td>
</tr>
</tbody>
</table>

- **BackupDirectoryWriteableCheck [339]**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BackupDirectoryWriteableCheck [339]</td>
<td>Checks that the configured backup directory is writeable</td>
</tr>
</tbody>
</table>

Confirms that the directory defined in `--backup-dir` directory exists and can be written to.

- **BackupDumpDirectoryWriteableCheck [339]**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BackupDumpDirectoryWriteableCheck [339]</td>
<td>Checks the backup temp directory is writeable</td>
</tr>
</tbody>
</table>

Confirms that the directory defined in `--backup-dump-dir` directory exists and can be written to.

- **BackupScriptAvailableCheck [339]**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BackupScriptAvailableCheck [339]</td>
<td>Checks that the configured backup script exists and can be executed</td>
</tr>
</tbody>
</table>

Confirms that the script defined in `--backup-script [367]` exists and is executable.

- **ClusterDiagnosticCheck [339]**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClusterDiagnosticCheck [339]</td>
<td></td>
</tr>
</tbody>
</table>

- **ClusterStatusCheck [339]**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClusterStatusCheck [339]</td>
<td></td>
</tr>
</tbody>
</table>

- **CommitDirectoryCheck [339]**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CommitDirectoryCheck [339]</td>
<td></td>
</tr>
</tbody>
</table>
## The tpm Deployment Command

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>- <strong>ConfigurationStorageDirectoryCheck</strong> [340]</td>
</tr>
<tr>
<td><strong>Option</strong></td>
</tr>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>- <strong>ConfigureValidationCheck</strong> [340]</td>
</tr>
<tr>
<td><strong>Option</strong></td>
</tr>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>- <strong>ConfiguredDirectoryCheck</strong> [340]</td>
</tr>
<tr>
<td><strong>Option</strong></td>
</tr>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>- <strong>ConflictingReplicationServiceTHLPortsCheck</strong> [340]</td>
</tr>
<tr>
<td><strong>Option</strong></td>
</tr>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>- <strong>ConnectorChecks</strong> [340]</td>
</tr>
<tr>
<td><strong>Option</strong></td>
</tr>
<tr>
<td><strong>Description</strong></td>
</tr>
</tbody>
</table>

Checks that the list of connectors and the corresponding list of data services is valid.

| - **ConnectorDBVersionCheck** [340] |
| **Option** | ConnectorDBVersionCheck [340] |
| **Description** |

| - **ConnectorListenerAddressCheck** [340] |
| **Option** | ConnectorListenerAddressCheck [340] |
| **Description** |

| - **ConnectorRWOAddressesCheck** [340] |
| **Option** | ConnectorRWOAddressesCheck [340] |
| **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** [340] |
| **Option** | ConnectorSmartScaleAllowedCheck [340] |
| **Description** | Confirms whether SmartScale is valid within the current configured parameters |

Checks that both SmartScale and Read/Write splitting have been enabled.

| - **ConnectorUserCheck** [340] |
| **Option** | ConnectorUserCheck [340] |
## The tpm Deployment Command

<table>
<thead>
<tr>
<th>Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>- ConsistentReplicationCredentialsCheck [341]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>ConsistentReplicationCredentialsCheck [341]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>- CurrentCommandCoordinatorCheck [341]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>CurrentCommandCoordinatorCheck [341]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>- CurrentConnectorCheck [341]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>CurrentConnectorCheck [341]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>- CurrentReleaseDirectoryIsSymlink [341]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>CurrentReleaseDirectoryIsSymlink [341]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>- CurrentTopologyCheck [341]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>CurrentTopologyCheck [341]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>- CurrentVersionCheck [341]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>CurrentVersionCheck [341]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>- DatasourceBootScriptCheck [341]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>DatasourceBootScriptCheck [341]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>- DifferentMasterSlaveCheck [341]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>DifferentMasterSlaveCheck [341]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>- DirectOracleServiceSIDCheck [341]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>DirectOracleServiceSIDCheck [341]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>- ElasticsearchValidationCheck [341]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>ElasticsearchValidationCheck [341]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>- EncryptionCheck [341]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>EncryptionCheck [341]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>Option</td>
</tr>
<tr>
<td>------------------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>EncryptionKeystoreCheck [342]</td>
<td>EncryptionKeystoreCheck [342]</td>
</tr>
<tr>
<td>FileValidationCheck [342]</td>
<td>FileValidationCheck [342]</td>
</tr>
<tr>
<td>FirewallCheck [342]</td>
<td>FirewallCheck [342]</td>
</tr>
<tr>
<td>GlobalHostAddressesCheck [342]</td>
<td>GlobalHostAddressesCheck [342]</td>
</tr>
<tr>
<td>GlobalMatchingPingMethodCheck [342]</td>
<td>GlobalMatchingPingMethodCheck [342]</td>
</tr>
<tr>
<td>GlobalRestartComponentsCheck [342]</td>
<td>GlobalRestartComponentsCheck [342]</td>
</tr>
<tr>
<td>GroupValidationCheck [342]</td>
<td>GroupValidationCheck [342]</td>
</tr>
<tr>
<td>HdfsValidationCheck [342]</td>
<td>HdfsValidationCheck [342]</td>
</tr>
<tr>
<td>HostLicensesCheck [342]</td>
<td>HostLicensesCheck [342]</td>
</tr>
<tr>
<td>HostOracleLibrariesFoundCheck [342]</td>
<td>HostOracleLibrariesFoundCheck [342]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>HostReplicatorServiceRunningCheck</strong> [343]</td>
<td></td>
</tr>
<tr>
<td><strong>Option</strong> HostReplicatorServiceRunningCheck [343]</td>
<td></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>HostSkippedChecks</strong> [343]</td>
<td></td>
</tr>
<tr>
<td><strong>Option</strong> HostSkippedChecks [343]</td>
<td></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>HostnameCheck</strong> [343]</td>
<td></td>
</tr>
<tr>
<td><strong>Option</strong> HostnameCheck [343]</td>
<td></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>HostsFileCheck</strong> [343]</td>
<td></td>
</tr>
<tr>
<td><strong>Option</strong> HostsFileCheck [343]</td>
<td></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>InstallServicesCheck</strong> [343]</td>
<td></td>
</tr>
<tr>
<td><strong>Option</strong> InstallServicesCheck [343]</td>
<td></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>InstallationScriptCheck</strong> [343]</td>
<td></td>
</tr>
<tr>
<td><strong>Option</strong> InstallationScriptCheck [343]</td>
<td></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>InstallerMasterSlaveCheck</strong> [343]</td>
<td></td>
</tr>
<tr>
<td><strong>Option</strong> InstallerMasterSlaveCheck [343]</td>
<td></td>
</tr>
<tr>
<td><strong>Description</strong> Checks whether a master host has been defined for the configured service.</td>
<td></td>
</tr>
<tr>
<td><strong>InstallingOverExistingInstallation</strong> [343]</td>
<td></td>
</tr>
<tr>
<td><strong>Option</strong> InstallingOverExistingInstallation [343]</td>
<td></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>JavaUserTimezoneCheck</strong> [343]</td>
<td></td>
</tr>
<tr>
<td><strong>Option</strong> JavaUserTimezoneCheck [343]</td>
<td></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>JavaVersionCheck</strong> [343]</td>
<td></td>
</tr>
<tr>
<td><strong>Option</strong> JavaVersionCheck [343]</td>
<td></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td><strong>KeystoresCheck</strong> [343]</td>
<td></td>
</tr>
<tr>
<td><strong>Option</strong> KeystoresCheck [343]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>Option</td>
</tr>
<tr>
<td>-------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>- KeystoresToCommitCheck [344]</td>
<td>KeystoresToCommitCheck [344]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>- ManagerActiveWitnessConversionCheck [344]</td>
<td>ManagerActiveWitnessConversionCheck [344]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>- ManagerChecks [344]</td>
<td>ManagerChecks [344]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>- ManagerHeapThresholdCheck [344]</td>
<td>ManagerHeapThresholdCheck [344]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>- ManagerListenerAddressCheck [344]</td>
<td>ManagerListenerAddressCheck [344]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>- ManagerPingMethodCheck [344]</td>
<td>ManagerPingMethodCheck [344]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>- ManagerWitnessAvailableCheck [344]</td>
<td>ManagerWitnessAvailableCheck [344]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>- ManagerWitnessNeededCheck [344]</td>
<td>ManagerWitnessNeededCheck [344]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>- MissingReplicationServiceConfigurationCheck [344]</td>
<td>MissingReplicationServiceConfigurationCheck [344]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>- ModifiedConfigurationFilesCheck [344]</td>
<td>ModifiedConfigurationFilesCheck [344]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
</tbody>
</table>
### Description

<table>
<thead>
<tr>
<th>Option</th>
<th>MySQLAllowIntensiveChecks [345]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Enables searching MySQL INFORMATION_SCHEMA for validation checks</td>
</tr>
</tbody>
</table>

Enables tpdm 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

<table>
<thead>
<tr>
<th>Option</th>
<th>MySQLApplierLogsCheck [345]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td></td>
</tr>
</tbody>
</table>

- MySQLApplierLogsCheck [345]

<table>
<thead>
<tr>
<th>Option</th>
<th>MySQLApplierPortCheck [345]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td></td>
</tr>
</tbody>
</table>

- MySQLApplierPortCheck [345]

<table>
<thead>
<tr>
<th>Option</th>
<th>MySQLApplierServerIDCheck [345]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td></td>
</tr>
</tbody>
</table>

- MySQLApplierServerIDCheck [345]

<table>
<thead>
<tr>
<th>Option</th>
<th>MySQLAvailableCheck [345]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Checks if MySQL is installed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>MySQLBinaryLogsEnabledCheck [345]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Checks that binary logging has been enabled on MySQL</td>
</tr>
</tbody>
</table>

Examines the log_bin variable has been defined within the running MySQL server. Binary logging must be enabled for replication to work.

<table>
<thead>
<tr>
<th>Option</th>
<th>MySQLBinlogDoDbCheck [345]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>MySQLClientCheck [345]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Checks whether the MySQL client command tool is available</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>MySQLConfigFileCheck [345]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Checks the existence of a MySQL configuration file</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>MySQLConnectorBridgeModePermissionsCheck [345]</th>
</tr>
</thead>
</table>
## The tpm Deployment Command

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQLConnectorBridgeModePermissionsCheck</td>
<td></td>
</tr>
<tr>
<td>MySQLConnectorPermissionsCheck</td>
<td></td>
</tr>
<tr>
<td>MySQLDefaultTableTypeCheck</td>
<td>Checks that the default table type configured for MySQL is a compatible transactional storage engine such as InnoDB</td>
</tr>
<tr>
<td>MySQLDumpCheck</td>
<td>Checks whether the <code>mysqldump</code> command within the configured <code>PATH</code> 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 <code>PATH</code> or use <code>--preferred-path</code> to point to the correct MySQL installation.</td>
</tr>
<tr>
<td>MySQLGeneratedColumnCheck</td>
<td>Checks, whether any tables contain generated or virtual columns. The test is only executed on MySQL 5.7 and only if <code>--mysql-allow-intensive-checks</code> has been enabled.</td>
</tr>
<tr>
<td>MySQLInnoDBEnabledCheck</td>
<td></td>
</tr>
<tr>
<td>MySQLJsonDataTypeCheck</td>
<td>Checks, whether any tables contain JSON columns. The test is only executed on MySQL 5.7 and only if <code>--mysql-allow-intensive-checks</code> has been enabled.</td>
</tr>
<tr>
<td>MySQLLoadDataInfilePermissionsCheck</td>
<td></td>
</tr>
<tr>
<td>MySQLLoginCheck</td>
<td>Checks whether Tungsten Clustering can connect to MySQL using the configured credentials</td>
</tr>
<tr>
<td>MySQLMyISAMCheck</td>
<td></td>
</tr>
</tbody>
</table>
### The tpm Deployment Command

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MySQLMyISAMCheck</strong> [346]</td>
<td>Checks for the existence of MyISAM tables</td>
</tr>
</tbody>
</table>

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 contain 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` 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`.

If neither of these methods is available, the check will fail and installation will stop.

- **MySQLNoMySQLReplicationCheck [347]**

- **MySQLPasswordSettingCheck [347]**

- **MySQLPermissionsCheck [347]**

- **MySQLReadableLogsCheck [347]**

- **MySQLSettingsCheck [347]**

- **MySQLSuperReadOnlyCheck [347]**

- **MySQLTriggerCheck [347]**

- **MySQLUnsupportedDataTypesCheck [347]**

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.
The tpm Deployment Command

<table>
<thead>
<tr>
<th>Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MysqlConnectorCheck [348]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>MysqlConnectorCheck [348]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>MysqlDumpAvailableCheck [348]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>MysqlDumpAvailableCheck [348]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>MysqlDumpSettingsCheck [348]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>MysqlDumpSettingsCheck [348]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>NewDirectoryRequiredCheck [348]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>NewDirectoryRequiredCheck [348]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>NtpdRunningCheck [348]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>NtpdRunningCheck [348]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>OSCheck [348]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>OSCheck [348]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>OldServicesRunningCheck [348]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>OldServicesRunningCheck [348]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>OpenFilesLimitCheck [348]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>OpenFilesLimitCheck [348]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>OpenSslLibraryCheck [348]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>OpenSslLibraryCheck [348]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>OracleLoginCheck [348]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>OracleLoginCheck [348]</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>OraclePermissionsCheck [348]</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>OraclePermissionsCheck [348]</td>
</tr>
</tbody>
</table>
### Description

- **OracleRedoReaderMinerDirectoryCheck [349]**

  **Option**: OracleRedoReaderMinerDirectoryCheck [349]

  **Description**

- **OracleServiceSIDCheck [349]**

  **Option**: OracleServiceSIDCheck [349]

  **Description**

- **OracleVersionCheck [349]**

  **Option**: OracleVersionCheck [349]

  **Description**

- **PGAAvailableCheck [349]**

  **Option**: PGAAvailableCheck [349]

  **Description**

- **ParallelReplicationCheck [349]**

  **Option**: ParallelReplicationCheck [349]

  **Description**

- **ParallelReplicationCountCheck [349]**

  **Option**: ParallelReplicationCountCheck [349]

  **Description**

- **PgControlAvailableCheck [349]**

  **Option**: PgControlAvailableCheck [349]

  **Description**

- **PgStandbyAvailableCheck [349]**

  **Option**: PgStandbyAvailableCheck [349]

  **Description**

- **PgdumpAvailableCheck [349]**

  **Option**: PgdumpAvailableCheck [349]

  **Description**

- **PgdumpallAvailableCheck [349]**

  **Option**: PgdumpallAvailableCheck [349]

  **Description**

- **PingSyntaxCheck [349]**

  **Option**: PingSyntaxCheck [349]
<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PortAvailabilityCheck [350]</td>
</tr>
<tr>
<td>Option</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td>ProfileScriptCheck [350]</td>
</tr>
<tr>
<td>Option</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td>RMIListenerAddressCheck [350]</td>
</tr>
<tr>
<td>Option</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td>RelayDirectoryWriteableCheck [350]</td>
</tr>
<tr>
<td>Option</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td>Confirms that the directory defined in \texttt{--relay-log-dir} directory exists and can be written to.</td>
</tr>
<tr>
<td>ReplicatorChecks [350]</td>
</tr>
<tr>
<td>Option</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td>RestartComponentsCheck [350]</td>
</tr>
<tr>
<td>Option</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td>RouterAffinityCheck [350]</td>
</tr>
<tr>
<td>Option</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td>RouterBridgeModeDefaultCheck [350]</td>
</tr>
<tr>
<td>Option</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td>RouterDelayBeforeOfflineCheck [350]</td>
</tr>
<tr>
<td>Option</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td>RouterKeepAliveTimeoutCheck [350]</td>
</tr>
<tr>
<td>Option</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td>RowBasedBinaryLoggingCheck [350]</td>
</tr>
<tr>
<td>Option</td>
</tr>
</tbody>
</table>
The `tpm` Deployment Command

<table>
<thead>
<tr>
<th>Description</th>
<th>Checks that Row-based binary logging has been enabled for heterogeneous deployments</th>
</tr>
</thead>
</table>

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` or `--enable-batch-service`, row-based logging within MySQL must have been switched on. The test looks for the value of `binlog_format=ROW`.

### RsyncAvailableCheck [351]

<table>
<thead>
<tr>
<th>Option</th>
<th>RsyncAvailableCheck [351]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td></td>
</tr>
</tbody>
</table>

### RubyVersionCheck [351]

<table>
<thead>
<tr>
<th>Option</th>
<th>RubyVersionCheck [351]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td></td>
</tr>
</tbody>
</table>

### SSHLoginCheck [351]

<table>
<thead>
<tr>
<th>Option</th>
<th>SSHLoginCheck [351]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Checks connectivity to other hosts over SSH</td>
</tr>
</tbody>
</table>

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
  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 [351]

<table>
<thead>
<tr>
<th>Option</th>
<th>ServiceTransferredLogStorageCheck [351]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td></td>
</tr>
</tbody>
</table>

### StartingStoppedServices [351]

<table>
<thead>
<tr>
<th>Option</th>
<th>StartingStoppedServices [351]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td></td>
</tr>
</tbody>
</table>

### SudoCheck [351]

<table>
<thead>
<tr>
<th>Option</th>
<th>SudoCheck [351]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td></td>
</tr>
</tbody>
</table>

### SwappinessCheck [351]

<table>
<thead>
<tr>
<th>Option</th>
<th>SwappinessCheck [351]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Checks the swappiness OS configuration is within a recommended range</td>
</tr>
</tbody>
</table>

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 Clustering when running. Change the value in `/sysctl.conf`.
### The `tpm` Deployment Command

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>THLDirectoryWriteableCheck</code></td>
<td>Ensures that the existing THL format is compatible with the new release.</td>
</tr>
<tr>
<td><code>THLListenerAddressCheck</code></td>
<td>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.</td>
</tr>
<tr>
<td><code>THLSchemaChangeCheck</code></td>
<td>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.</td>
</tr>
<tr>
<td><code>THLStorageCheck</code></td>
<td>Confirms the THL storage directory exists, is empty and writeable.</td>
</tr>
<tr>
<td><code>THLStorageChecksum</code></td>
<td></td>
</tr>
<tr>
<td><code>TargetDirectoryDoesNotExist</code></td>
<td></td>
</tr>
<tr>
<td><code>TransferredLogStorageCheck</code></td>
<td></td>
</tr>
<tr>
<td><code>UpgradeSameProductCheck</code></td>
<td>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.</td>
</tr>
<tr>
<td><code>VIPEnabledHostAllowsRootCommands</code></td>
<td></td>
</tr>
<tr>
<td><code>VIPEnabledHostArpPath</code></td>
<td></td>
</tr>
</tbody>
</table>
### Description

<table>
<thead>
<tr>
<th>Option</th>
<th>VIPEnabledHostIfconfigPath [353]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>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.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>VerticaUserGroupsCheck [353]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Checks that the Vertica user has the correct OS group membership</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>WhichAvailableCheck [353]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Checks the existence of a working which command</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>WriteableHomeDirectoryCheck [353]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Checks that the home directory for the configured user can be written to</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>WriteableTempDirectoryCheck [353]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Ensures the temporary directory can be written to</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>XtrabackupAvailableCheck [353]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>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.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>XtrabackupDirectoryWriteableCheck [353]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>XtrabackupSettingsCheck [353]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td></td>
</tr>
</tbody>
</table>

### 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 [403]`
- In an INI file, without the double-dash prefix, i.e. `repl-thl-log-retention=3d [403]`
A full list of all the available options supported is provided in Table 9.6, “tpm Configuration Options”.

<table>
<thead>
<tr>
<th>CommandLine Option</th>
<th>INI File Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--allow-bidi-unsafe [363], --rep-pl-allow-bidi-unsafe [363]</td>
<td>allow-bidi-unsafe [363], repl-allow-bidi-unsafe [363]</td>
<td>Allow unsafe SQL from remote service</td>
</tr>
<tr>
<td>--api [363], --rep-pl-api [363]</td>
<td>api [363], repl-api [363]</td>
<td>Enable the replication API</td>
</tr>
<tr>
<td>--api-host [363], --rep-pl-api-host [363]</td>
<td>api-host [363], repl-api-host [363]</td>
<td>Hostname that the replication API should listen on</td>
</tr>
<tr>
<td>--api-port [364], --rep-pl-api-port [364]</td>
<td>api-port [364], repl-api-port [364]</td>
<td>Port that the replication API should bind to</td>
</tr>
<tr>
<td>--api-user [364], --rep-pl-api-user [364]</td>
<td>api-user [364], repl-api-user [364]</td>
<td>HTTP basic auth username for the replication API</td>
</tr>
<tr>
<td>--application-password [364], --connector-password [364]</td>
<td>application-password [364], connector-password [364]</td>
<td>Database password for the connector</td>
</tr>
<tr>
<td>--application-port [364], --connector-listen-port [364]</td>
<td>application-port [364], connector-listen-port [364]</td>
<td>Port for the connector to listen on</td>
</tr>
<tr>
<td>--application-readonly-port [364], --connector-readonly-listen-port [364]</td>
<td>application-readonly-port [364], connector-readonly-listen-port [364]</td>
<td>Port for the connector to listen for read-only connections on</td>
</tr>
<tr>
<td>--application-user [365], --connector-user [365]</td>
<td>application-user [365], connector-user [365]</td>
<td>Database username for the connector</td>
</tr>
<tr>
<td>--auto-enable [365], --rep-pl-auto-enable [365]</td>
<td>auto-enable [365], repl-auto-enable [365]</td>
<td>Auto-enable services after start-up</td>
</tr>
<tr>
<td>--auto-recovery-reset-interval [365], --rep-pl-auto-recovery-reset-interval [365]</td>
<td>auto-recovery-reset-interval [365], repl-auto-recovery-reset-interval [365]</td>
<td>Delay before autorecovery is deemed to have succeeded</td>
</tr>
<tr>
<td>--backup-online [366], --rep-pl-backup-online [366]</td>
<td>backup-online [366], repl-backup-online [366]</td>
<td>Does the backup script support backing up a datasource while it is ONLINE</td>
</tr>
<tr>
<td>--backup-retention [366], --rep-pl-backup-retention [366]</td>
<td>backup-retention [366], repl-backup-retention [366]</td>
<td>Number of backups to retain</td>
</tr>
<tr>
<td>--backup-script [367], --rep-pl-backup-script [367]</td>
<td>backup-script [367], repl-backup-script [367]</td>
<td>What is the path to the backup script</td>
</tr>
<tr>
<td>--batch-enabled [367]</td>
<td>batch-enabled [367]</td>
<td>Should the replicator service use a batch applier</td>
</tr>
<tr>
<td>--batch-load-template [367]</td>
<td>batch-load-template [367]</td>
<td>Value for the loadBatchTemplate property</td>
</tr>
<tr>
<td>CmdLine Option</td>
<td>INI File Option</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>--channels [367], --repl-channels [367]</td>
<td>channels [367], repl-channels [367]</td>
<td>Number of replication channels to use for services</td>
</tr>
<tr>
<td>--cluster-slave-auto-recovery-max-attempts [368], --cluster-slave-repl-auto-recovery-max-attempts [368]</td>
<td>cluster-slave-auto-recovery-max-attempts [368], cluster-slave-repl-auto-recovery-max-attempts [368]</td>
<td>Default value for --auto-recovery-max-attempts when --topology=cluster-slave</td>
</tr>
<tr>
<td>--composite-datasources [368], --dataservice-composite-datasources [368]</td>
<td>composite-datasources [368], dataservice-composite-datasources [368]</td>
<td>Data services that should be added to this composite data service</td>
</tr>
<tr>
<td>--config-file-help [368]</td>
<td>config-file-help [368]</td>
<td>Display help information for content of the config file</td>
</tr>
<tr>
<td>--conn-java-enable-concurrency [368]</td>
<td>conn-java-enable-concurrency [368]</td>
<td>Connector Java uses concurrent garbage collection</td>
</tr>
<tr>
<td>--conn-java-mem-size [368]</td>
<td>conn-java-mem-size [368]</td>
<td>Connector Java heap memory size used to buffer data between clients and databases</td>
</tr>
<tr>
<td>--conn-round-robin-include-master [368]</td>
<td>conn-round-robin-include-master [368]</td>
<td>Should the Connector include the master in round-robin load balancing</td>
</tr>
<tr>
<td>--connector-affinity [369]</td>
<td>connector-affinity [369]</td>
<td>The default affinity for all connections</td>
</tr>
<tr>
<td>--connector-autoreconnect [369]</td>
<td>connector-autoreconnect [369]</td>
<td>Enable auto-reconnect in the connector</td>
</tr>
<tr>
<td>--connector-autoreconnect-killed-connections [369]</td>
<td>connector-autoreconnect-killed-connections [369]</td>
<td>Enable autoreconnect for connections killed within the connector</td>
</tr>
<tr>
<td>--connector-bridge-mode [369], --enable-connector-bridge-mode [369]</td>
<td>connector-bridge-node [369], enable-connector-bridge-mode [369]</td>
<td>Enable the Tungsten Connector bridge mode</td>
</tr>
<tr>
<td>--connector-delete-user-map [369]</td>
<td>connector-delete-user-map [369]</td>
<td>Overwrite an existing user.map file</td>
</tr>
<tr>
<td>--connector-drop-after-max-connections [370]</td>
<td>connector-drop-after-max-connections [370]</td>
<td>Instantly drop connections that arrive after --connector-max-connections has been reached</td>
</tr>
<tr>
<td>--connector-keepalive-timeout [370]</td>
<td>connector-keepalive-timeout [370]</td>
<td>The Connector keep-alive SELECT statement is submitted if the time since the last activity reaches this timeout value.</td>
</tr>
<tr>
<td>--connector-listen-interface [370]</td>
<td>connector-listen-interface [370]</td>
<td>Listen interface to use for the connector</td>
</tr>
<tr>
<td>--connector-max-connections [370]</td>
<td>connector-max-connections [370]</td>
<td>The maximum number of connections the connector should allow at any time</td>
</tr>
<tr>
<td>--connector-max-slave-latency [370], --connector-max-applied-latency [370]</td>
<td>connector-max-applied-latency [370], connector-max-slave-latency [370]</td>
<td>The maximum applied latency for slave connections</td>
</tr>
<tr>
<td>--connector-readonly [370], --enable-connector-readonly [370]</td>
<td>connector-readonly [370], enable-connector-readonly [370]</td>
<td>Enable the Tungsten Connector read-only mode</td>
</tr>
<tr>
<td>--connector-ro-addresses [371]</td>
<td>connector-ro-addresses [371]</td>
<td>Connector addresses that should receive a r/o connection</td>
</tr>
<tr>
<td>--connector-rw-addresses [371]</td>
<td>connector-rw-addresses [371]</td>
<td>Connector addresses that should receive a r/w connection</td>
</tr>
<tr>
<td>--connector-smartscale [371]</td>
<td>connector-smartscale [371]</td>
<td>Enable SmartScale R/W splitting in the connector</td>
</tr>
<tr>
<td>CmdLine Option</td>
<td>INI File Option</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>--connector-smartscale-session-id [371]</code></td>
<td><code>connector-smartscale-session-id [371]</code></td>
<td>The default session ID to use with smart scale</td>
</tr>
<tr>
<td><code>--consistency-policy [371], --repl-consistency-policy [371]</code></td>
<td><code>consistency-policy [371], repl-consistency-policy [371]</code></td>
<td>Should the replator stop or warn if a consistency check fails?</td>
</tr>
<tr>
<td><code>--dataset-service-name [372]</code></td>
<td><code>dataset-service-name [372]</code></td>
<td>Limit the command to the hosts in this dataset-service Multiple data services may be specified by providing a comma separated list</td>
</tr>
<tr>
<td><code>--dataset-service-relay-enabled [372]</code></td>
<td><code>dataset-service-relay-enabled [372]</code></td>
<td>Make this dataset-service the slave of another</td>
</tr>
<tr>
<td><code>--dataset-service-schema [372]</code></td>
<td><code>dataset-service-schema [372]</code></td>
<td>The db schema to hold dataset-service details</td>
</tr>
<tr>
<td><code>--dataset-service-thl-port [372]</code></td>
<td><code>dataset-service-thl-port [372]</code></td>
<td>Port to use for THL operations</td>
</tr>
<tr>
<td><code>--dataset-service-use-relative-latency [372], --use-relative-latency [372]</code></td>
<td><code>dataset-service-use-relative-latency [372], use-relative-latency [372]</code></td>
<td>Enable the cluster to operate on relative latency</td>
</tr>
<tr>
<td><code>--dataset-service-vip-enabled [372]</code></td>
<td><code>dataset-service-vip-enabled [372]</code></td>
<td>Is VIP management enabled?</td>
</tr>
<tr>
<td><code>--dataset-service-vip-ipaddress [372]</code></td>
<td><code>dataset-service-vip-ipaddress [372]</code></td>
<td>VIP IP address</td>
</tr>
<tr>
<td><code>--dataset-source-boot-script [373], repl-datasource-boot-script [373]</code></td>
<td><code>dataset-source-boot-script [373], repl-datasource-boot-script [373]</code></td>
<td>Database start script</td>
</tr>
<tr>
<td><code>--dataset-source-enable-ssl [373], repl-datasource-enable-ssl [373]</code></td>
<td><code>dataset-source-enable-ssl [373], repl-datasource-enable-ssl [373]</code></td>
<td>Enable SSL connection to DBMS server</td>
</tr>
<tr>
<td><code>--dataset-source-oracle-service-group [375], repl-datasource-oracle-service-group [375]</code></td>
<td><code>dataset-source-oracle-service-group [375], repl-datasource-oracle-service-group [375]</code></td>
<td>Oracle service operating system group</td>
</tr>
</tbody>
</table>
**The tpm Deployment Command**

<table>
<thead>
<tr>
<th>CmdLine Option</th>
<th>INI File Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>repl-datasource-archive [375], repl-datasource-pg-home [375]</td>
<td>datasource-archive [375], repl-datasource-archive [375]</td>
<td>PostgreSQL archive location</td>
</tr>
<tr>
<td>repl-datasource-conf [375], repl-datasource-pg-conf [375]</td>
<td>datasource-conf [375], repl-datasource-conf [375]</td>
<td>Location of postgresql.conf</td>
</tr>
<tr>
<td>repl-datasource-home [375], repl-datasource-pg-home [375]</td>
<td>datasource-home [375], repl-datasource-pg-home [375]</td>
<td>PostgreSQL data directory</td>
</tr>
<tr>
<td>repl-datasource-systemctl-service [375], repl-datasource-systemctl-service [375]</td>
<td>datasource-systemctl-service [375], repl-datasource-systemctl-service [375]</td>
<td>Database systemctl script</td>
</tr>
<tr>
<td>repl-datasource-type [375], repl-datasource-type [375]</td>
<td>datasource-type [375], repl-datasource-type [375]</td>
<td>Database type</td>
</tr>
<tr>
<td>repl-datasource-user [375], repl-datasource-user [375]</td>
<td>datasource-user [375], repl-datasource-user [375]</td>
<td>Database login for Tungsten</td>
</tr>
<tr>
<td>repl-datasource-password [375], repl-datasource-password [375]</td>
<td>datasource-password [375], repl-datasource-password [375]</td>
<td>Password for datasource connection</td>
</tr>
<tr>
<td>repl-datasource-port [375], repl-datasource-port [375]</td>
<td>datasource-port [375], repl-datasource-port [375]</td>
<td>Database server port</td>
</tr>
<tr>
<td>repl-datasource-user [375], repl-datasource-user [375]</td>
<td>datasource-user [375], repl-datasource-user [375]</td>
<td>Database login for Tungsten</td>
</tr>
<tr>
<td>directory [378]</td>
<td>directory [378]</td>
<td>Set the directory of an existing installation used during fetching an existing configuration</td>
</tr>
<tr>
<td>repl-datasource-password [378], repl-datasource-password [378]</td>
<td>datasource-password [378], repl-datasource-password [378]</td>
<td>Disable the use of relay-logs?</td>
</tr>
<tr>
<td>repl-datasource-user [378], repl-datasource-user [378]</td>
<td>datasource-user [378], repl-datasource-user [378]</td>
<td>Should slave servers support the master role?</td>
</tr>
<tr>
<td>repl-datasource-password [378], repl-datasource-password [378]</td>
<td>datasource-password [378], repl-datasource-password [378]</td>
<td>Disables all forms of security, including SSL, TLS and authentication</td>
</tr>
<tr>
<td>repl-datasource-user [378], repl-datasource-user [378]</td>
<td>datasource-user [378], repl-datasource-user [378]</td>
<td>This will modify UPDATE transactions in row-based replication and eliminate any columns that were not modified.</td>
</tr>
<tr>
<td>repl-datasource-password [378], repl-datasource-password [378]</td>
<td>datasource-password [378], repl-datasource-password [378]</td>
<td></td>
</tr>
<tr>
<td>CmdLine Option</td>
<td>INI File Option</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>--enable-active-witnesses [378], --active-witnesses [378]</td>
<td>active-witnesses [378], enable-active-witnesses [378]</td>
<td>Enable active witness hosts</td>
</tr>
<tr>
<td>--enable-batch-master [378]</td>
<td>enable-batch-master [378]</td>
<td>Enable batch operation for the master</td>
</tr>
<tr>
<td>--enable-batch-service [379]</td>
<td>enable-batch-service [379]</td>
<td>Enables batch mode for a service</td>
</tr>
<tr>
<td>--enable-batch-slae [379]</td>
<td>enable-batch-slae [379]</td>
<td>Enable batch operation for the slave</td>
</tr>
<tr>
<td>--enable-connector-client-ssl [379], --connector-client-ssl [379]</td>
<td>connector-client-ssl [379], enable-connector-client-ssl [379]</td>
<td>Enable SSL encryption of traffic from the client to the connector</td>
</tr>
<tr>
<td>--enable-connector-server-ssl [379], --connector-server-ssl [379]</td>
<td>connector-server-ssl [379], enable-connector-server-ssl [379]</td>
<td>Enable SSL encryption of traffic from the connector to the database</td>
</tr>
<tr>
<td>--enable-connector-ssl [379], --connector-ssl [379]</td>
<td>connector-ssl [379], enable-connector-ssl [379]</td>
<td>Enable SSL encryption of connector traffic to the database</td>
</tr>
<tr>
<td>--enable-heterogeneous-master [380], --enable-heterogeneous-master [380]</td>
<td>enable-heterogeneous-master [380], enable-heterogeneous-master [380]</td>
<td>Enable heterogeneous operation for the master</td>
</tr>
<tr>
<td>--enable-heterogeneous-service [380], --enable-heterogeneous-service [380]</td>
<td>enable-heterogeneous-service [380], enable-heterogeneous-service [380]</td>
<td>Enable heterogeneous operation</td>
</tr>
<tr>
<td>--enable-heterogeneous-slave [380], --enable-heterogeneous-slave [380]</td>
<td>enable-heterogeneous-slave [380], enable-heterogeneous-slave [380]</td>
<td>Enable heterogeneous operation for the slave</td>
</tr>
<tr>
<td>--enable-jgroups-ssl [380], --jgroups-ssl [380]</td>
<td>enable-jgroups-ssl [380], jgroups-ssl [380]</td>
<td>Enable SSL encryption of JGroups communication on this host</td>
</tr>
<tr>
<td>--enable-rmi-authentication [380], --rmi-authentication [380]</td>
<td>enable-rmi-authentication [380], rmi-authentication [380]</td>
<td>Enable RMI authentication for the services running on this host</td>
</tr>
<tr>
<td>--enable-rmi-ssl [381], --rmi-ssl [381]</td>
<td>enable-rmi-ssl [381], rmi-ssl [381]</td>
<td>Enable SSL encryption of RMI communication on this host</td>
</tr>
<tr>
<td>--enable-slave-thl-listener [381], --repl-enable-slave-thl-listener [381]</td>
<td>enable-slave-thl-listener [381], repl-enable-slave-thl-listener [381]</td>
<td>Should this service allow THL connections?</td>
</tr>
<tr>
<td>--enable-sudo-access [381], --root-command-prefix [381]</td>
<td>enable-sudo-access [381], root-command-prefix [381]</td>
<td>Run root commands using sudo</td>
</tr>
<tr>
<td>--enable-thl-ssl [381], --repl-enable-thl-ssl [381], --thl-ssl [381]</td>
<td>enable-thl-ssl [381], repl-enable-thl-ssl [381], thl-ssl [381]</td>
<td>Enable SSL encryption of THL communication for this service</td>
</tr>
<tr>
<td>--executable-prefix [381]</td>
<td>executable-prefix [381]</td>
<td>Adds a prefix to command aliases</td>
</tr>
<tr>
<td>--host [382]</td>
<td>host-name [382]</td>
<td>DNS hostname</td>
</tr>
<tr>
<td>--hosts [382]</td>
<td>hosts [382]</td>
<td>Limit the command to the hosts listed You must use the hostname as it appears in the configuration.</td>
</tr>
<tr>
<td>--hub [382], --dataservice-hub-host [382]</td>
<td>dataservice-hub-host [382], hub-host [382]</td>
<td>What is the hub host for this all-masters dataservice?</td>
</tr>
<tr>
<td>--hub-service [382], --dataservice-service-host [382]</td>
<td>dataservice-service-host [382], hub-service [382]</td>
<td>The data service to use for the hub of a star topology</td>
</tr>
<tr>
<td>--install [382]</td>
<td>install [382]</td>
<td>Install service start scripts</td>
</tr>
<tr>
<td>--install-directory [382], --home-directory [382]</td>
<td>home-directory [382], install-directory [382]</td>
<td>Installation directory</td>
</tr>
<tr>
<td>--java-connector-key-store-path [383]</td>
<td>java-connector-key-store-path [383]</td>
<td>Local path to the Java Connector Keystore file.</td>
</tr>
<tr>
<td>CmdLine Option</td>
<td>INI File Option</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>-java-jgroups-key [384]</td>
<td>java-jgroups-key [384]</td>
<td>The alias to use for the JGroups TLS key in the keystore.</td>
</tr>
<tr>
<td>-java-jgroups-keystore-path [384]</td>
<td>java-jgroups-keystore-path [384]</td>
<td>Local path to the JGroups Java Keyfile store.</td>
</tr>
<tr>
<td>-java-tls-key-lifetime [385]</td>
<td>java-tls-key-lifetime [385]</td>
<td>Lifetime for the Java TLS key</td>
</tr>
<tr>
<td>-java-truststore-password [385]</td>
<td>java-truststore-password [385]</td>
<td>The password for unlocking the tungsten_truststore.jks file in the security directory</td>
</tr>
<tr>
<td>-java-user-timezone [385], repl-java-user-timezone [385]</td>
<td>java-user-timezone [385], repl-java-user-timezone [385]</td>
<td>Java VM Timezone (esp. for cross-site replication)</td>
</tr>
<tr>
<td>-log [385]</td>
<td>log [385]</td>
<td>Write all messages, visible and hidden, to this file. You may specify a filename, ‘pid’ or ‘timestamp’.</td>
</tr>
<tr>
<td>-log-slave-updates [385]</td>
<td>log-slave-updates [385]</td>
<td>Should slaves log updates to binlog</td>
</tr>
<tr>
<td>-master [386], -dataservice-master-host [386], -master-host [386], -meters [386], -relay [386]</td>
<td>dataservice-master-host [386], master-host [386], masters [386], relay [386]</td>
<td>Hostname of the master host within this service</td>
</tr>
<tr>
<td>-master-preferred-role [386], repl-master-preferred-role [386]</td>
<td>master-preferred-role [386], repl-master-preferred-role [386]</td>
<td>Preferred role for master THL when connecting as a slave</td>
</tr>
<tr>
<td>-master-services [386], -dataservice-master-services [386]</td>
<td>dataservice-master-services [386], master-services [386]</td>
<td>Data service names that should be used on each master</td>
</tr>
<tr>
<td>-members [386], -dataservice-hosts [386]</td>
<td>dataservice-hosts [386], members [386]</td>
<td>Hostnames for the dataservice members</td>
</tr>
<tr>
<td>-mgr-api [387]</td>
<td>mgr-api [387]</td>
<td>Enable the Manager API</td>
</tr>
<tr>
<td>-mgr-api-address [387]</td>
<td>mgr-api-address [387]</td>
<td>Address for the Manager API</td>
</tr>
<tr>
<td>-mgr-api-full-access [387]</td>
<td>mgr-api-full-access [387]</td>
<td>Enable all Manager API commands. Only the status command will be enabled without it.</td>
</tr>
<tr>
<td>-mgr-api-port [387]</td>
<td>mgr-api-port [387]</td>
<td>Port for the Manager API</td>
</tr>
<tr>
<td>CmdLine Option</td>
<td>INI File Option</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>repl-group-communication-port</td>
<td>mgr-group-communication-port</td>
<td>Port to use for manager group communication</td>
</tr>
<tr>
<td>repl-heap-threshold</td>
<td>mgr-heap-threshold</td>
<td>Java memory usage [MB] that will force a Manager restart</td>
</tr>
<tr>
<td>repl-java-enable-concurrent-gc</td>
<td>mgr-java-enable-concurrent-gc</td>
<td>Manager Java uses concurrent garbage collection</td>
</tr>
<tr>
<td>repl-java-heap-size</td>
<td>mgr-java-heap-size</td>
<td>Manager Java heap memory size in Mb [min 128]</td>
</tr>
<tr>
<td>repl-listen-interface</td>
<td>mgr-listen-interface</td>
<td>Listen interface to use for the manager</td>
</tr>
<tr>
<td>repl-ping-method</td>
<td>mgr-ping-method</td>
<td>Mechanism to use when identifying the liveness of other data-sources (ping, echo)</td>
</tr>
<tr>
<td>repl-policy-mode</td>
<td>mgr-policy-mode</td>
<td>Manager policy mode</td>
</tr>
<tr>
<td>repl-rmi-port</td>
<td>mgr-rmi-port</td>
<td>Port to use for the manager RMI server</td>
</tr>
<tr>
<td>repl-rmi-remote-port</td>
<td>mgr-rmi-remote-port</td>
<td>Port to use for calling the remote manager RMI server</td>
</tr>
<tr>
<td>repl-ro-slave</td>
<td>mgr-ro-slave</td>
<td>Make slaves read-only</td>
</tr>
<tr>
<td>repl-vip-arp-path</td>
<td>mgr-vip-arp-path</td>
<td>Path to the arp binary</td>
</tr>
<tr>
<td>repl-vip-device</td>
<td>mgr-vip-device</td>
<td>VIP network device</td>
</tr>
<tr>
<td>repl-vip-ifconfig-path</td>
<td>mgr-vip-ifconfig-path</td>
<td>Path to the ifconfig binary</td>
</tr>
<tr>
<td>repl-wait-for-members</td>
<td>mgr-wait-for-members</td>
<td>Wait for all datasources to be available before completing installation</td>
</tr>
<tr>
<td>mysql-allow-intensive-checks</td>
<td>mysql-allow-intensive-checks</td>
<td>For MySQL installation, enables detailed checks on the supported data types within the MySQL database to confirm compatibility.</td>
</tr>
<tr>
<td>mysql-connectorj-path</td>
<td>mysql-connectorj-path</td>
<td>Path to MySQL Connector/J</td>
</tr>
<tr>
<td>mysql-driver</td>
<td>mysql-driver</td>
<td>MySQL Driver Vendor</td>
</tr>
<tr>
<td>mysql-enable-ansiquotes</td>
<td>mysql-enable-ansiquotes</td>
<td>Enables ANSI_QUOTES mode for incoming events?</td>
</tr>
<tr>
<td>mysql-enable-enumtostring</td>
<td>mysql-enable-enumtostring</td>
<td>Enable a filter to convert ENUM values to strings</td>
</tr>
<tr>
<td>mysql-enable-noonlykeywords</td>
<td>mysql-enable-noonlykeywords</td>
<td>Enables a filter to translate DELETE FROM ONLY to DELETE FROM and UPDATE ONLY TO UPDATE.</td>
</tr>
<tr>
<td>mysql-enable-settostring</td>
<td>mysql-enable-settostring</td>
<td>Enable a filter to convert SET types to strings</td>
</tr>
<tr>
<td>mysql-heap-size</td>
<td>mysql-heap-size</td>
<td>Slaves are read-only</td>
</tr>
<tr>
<td>mysql-server-id</td>
<td>mysql-server-id</td>
<td>Explicitly set the MySQL server ID</td>
</tr>
<tr>
<td>mysql-use-bytes-for-string</td>
<td>mysql-use-bytes-for-string</td>
<td>Transfer strings as their byte representation?</td>
</tr>
<tr>
<td>mysql-xtrabackup-dir</td>
<td>mysql-xtrabackup-dir</td>
<td>Directory to use for storing xtrabackup full &amp; incremental backups</td>
</tr>
<tr>
<td>native-slave-takeover</td>
<td>native-slave-takeover</td>
<td>Takeover native replication</td>
</tr>
<tr>
<td>no-deployment</td>
<td>no-deployment</td>
<td>Skip deployment steps that create the install directory</td>
</tr>
<tr>
<td>no-validation</td>
<td>no-validation</td>
<td>Skip validation checks that run on each host</td>
</tr>
<tr>
<td>optimize-row-events</td>
<td>optimize-row-events</td>
<td>Enables or disabled optimized row updates</td>
</tr>
<tr>
<td>oracle-extractor-method</td>
<td>oracle-extractor-method</td>
<td>Oracle extractor method</td>
</tr>
<tr>
<td>pg-archive-timeout</td>
<td>pg-archive-timeout</td>
<td>Timeout for sending unfilled WAL buffers [data loss window]</td>
</tr>
<tr>
<td>CommandLine Option</td>
<td>Ini File Option</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>--pg-ctl [392], --reppl-pg-ctl [392]</td>
<td>pg-ctl [392], repl-pg-ctl [392]</td>
<td>Path to the pg_ctl script</td>
</tr>
<tr>
<td>--pg-standby [392], --reppl-pg-standby [392]</td>
<td>pg-standby [392], repl-pg-standby [392]</td>
<td>Path to the pg_stabndby script</td>
</tr>
<tr>
<td>--postgresql-dbname [392], --repl-postgresql-dbname [392]</td>
<td>postgresql-dbname [392], repl-postgresql-dbname [392]</td>
<td>Name of the database to replicate</td>
</tr>
<tr>
<td>--postgresql-enable-nysql2pgddl [393], --repl-postgresql-enable-nysql2pgddl [393]</td>
<td>postgresql-enable-nysql2pgddl [393], repl-postgresql-enable-nysql2pgddl [393]</td>
<td>Enable MySQL to PostgreSQL DDL dialect converting filter placeholders</td>
</tr>
<tr>
<td>--postgresql-slonik [393], --repl-postgresql-slonik [393]</td>
<td>postgresql-slonik [393], repl-postgresql-slonik [393]</td>
<td>Path to the slonik executable</td>
</tr>
<tr>
<td>--postgresql-tables [393], --repl-postgresql-tables [393]</td>
<td>postgresql-tables [393], repl-postgresql-tables [393]</td>
<td>Tables to replicate in form: schema1.table1,schema2.table2,...</td>
</tr>
<tr>
<td>--preferred-path [393]</td>
<td>preferred-path [393]</td>
<td>Additional command path</td>
</tr>
<tr>
<td>--prefetch-enabled [393]</td>
<td>prefetch-enabled [393]</td>
<td>Should the replicator service be setup as a prefetch applier</td>
</tr>
<tr>
<td>--prefetch-max-time-ahead [393]</td>
<td>prefetch-max-time-ahead [393]</td>
<td>Maximum number of seconds that the prefetch applier can get in front of the standard applier</td>
</tr>
<tr>
<td>--prefetch-min-time-ahead [394]</td>
<td>prefetch-min-time-ahead [394]</td>
<td>Minimum number of seconds that the prefetch applier must be in front of the standard applier</td>
</tr>
<tr>
<td>--prefetch-sleep-time [394]</td>
<td>prefetch-sleep-time [394]</td>
<td>How long to wait when the prefetch applier gets too far ahead</td>
</tr>
<tr>
<td>--privileged-master [394]</td>
<td>privileged-master [394]</td>
<td>Does the login for the master database service have superuser privileges</td>
</tr>
<tr>
<td>--privileged-slave [394]</td>
<td>privileged-slave [394]</td>
<td>Does the login for the slave database service have superuser privileges</td>
</tr>
<tr>
<td>--profile-script [394]</td>
<td>profile-script [394]</td>
<td>Append commands to include env.sh in this profile script</td>
</tr>
<tr>
<td>--protect-configuration-files [394]</td>
<td>protect-configuration-files [394]</td>
<td>When enabled, configuration files are protected to be only readable and updatable by the configured user</td>
</tr>
<tr>
<td>--redshift-dbname [395], --reppl-redshift-dbname [395]</td>
<td>redshift-dbname [395], repl-redshift-dbname [395]</td>
<td>Name of the Redshift database to replicate into</td>
</tr>
<tr>
<td>--relay-enabled [395]</td>
<td>relay-enabled [395]</td>
<td>Should the replicator service be setup as a relay master</td>
</tr>
<tr>
<td>--relay-source [396], --dataservice-relay-source [396], --master-dataservice [396]</td>
<td>dataservice-relay-source [396], master-dataservice [396], relay-source [396]</td>
<td>Dataservice name to use as a relay source</td>
</tr>
<tr>
<td>--replace-tls-certificate [396]</td>
<td>replace-tls-certificate [396]</td>
<td>Replace the TLS certificate</td>
</tr>
<tr>
<td>--replication-host [396], --datasource-host [396], --repl-datasource-host [396]</td>
<td>datasource-host [396], repl-datasource-host [396], replication-host [396]</td>
<td>Hostname of the datasource</td>
</tr>
<tr>
<td>--replication-password [396], --datasource-password [396], --repl-datasource-password [396]</td>
<td>datasource-password [396], repl-datasource-password [396], replication-password [396]</td>
<td>Database password</td>
</tr>
<tr>
<td>--replication-port [396], --datasource-port [396], --repl-datasource-port [396]</td>
<td>datasource-port [396], repl-datasource-port [396], replication-port [396]</td>
<td>Database network port</td>
</tr>
<tr>
<td>--replication-user [397], --datasource-user [397], --repl-datasource-user [397]</td>
<td>datasource-user [397], repl-datasource-user [397], replication-user [397]</td>
<td>User for database connection</td>
</tr>
<tr>
<td>--reset [397]</td>
<td>reset [397]</td>
<td>Clear the current configuration before processing any arguments</td>
</tr>
<tr>
<td>CmdLine Option</td>
<td>INI File Option</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>repl-rmi-port [397]</td>
<td>repl-rmi-port [397], rmi-port [397]</td>
<td>Replication RMI port</td>
</tr>
<tr>
<td>repl-rmi-user [397]</td>
<td>repl-rmi-user [397]</td>
<td>The username for RMI authentication</td>
</tr>
<tr>
<td>repl-role [397], repl-role [397]</td>
<td>repl-role [397], role [397]</td>
<td>What is the replication role for this service?</td>
</tr>
<tr>
<td>repl-router-gateway-port [397]</td>
<td>repl-router-gateway-port [397]</td>
<td>The router gateway port</td>
</tr>
<tr>
<td>repl-replication-service-type [398]</td>
<td>repl-replication-service-type [398]</td>
<td>What is the replication service type?</td>
</tr>
<tr>
<td>repl-service-alias [398], repl-service-alias [398]</td>
<td>repl-service-alias [398], service-alias [398]</td>
<td>Replication alias of this dataservice</td>
</tr>
<tr>
<td>repl-service-name [398]</td>
<td>repl-service-name [398], service-name [398]</td>
<td>Set the service name</td>
</tr>
<tr>
<td>repl-skip-statemap [398]</td>
<td>repl-skip-statemap [398]</td>
<td>Do not copy the cluster-home/conf/statemap.properties from the previous install</td>
</tr>
<tr>
<td>repl-slaves [398], repl-slaves [398]</td>
<td>repl-slaves [398],.dataservice-slaves [398], slaves [398]</td>
<td>What are the slaves for this dataservice?</td>
</tr>
<tr>
<td>repl-start [399]</td>
<td>repl-start [399]</td>
<td>Start the services after configuration</td>
</tr>
<tr>
<td>repl-start-and-report [399]</td>
<td>repl-start-and-report [399]</td>
<td>Start the services and report out the status after configuration</td>
</tr>
<tr>
<td>repl-svc-allow-any-remote-service [399], repl-svc-allow-any-remote-service [399]</td>
<td>repl-svc-allow-any-remote-service [399], svc-allow-any-remote-service [399]</td>
<td>Replicate from any service</td>
</tr>
<tr>
<td>repl-svc-applier-filters [399], repl-svc-applier-filters [399]</td>
<td>repl-svc-applier-filters [399], svc-applier-filters [399]</td>
<td>Replication service applier filters</td>
</tr>
<tr>
<td>repl-svc-extractor-filters [400], repl-svc-extractor-filters [400]</td>
<td>repl-svc-extractor-filters [400], svc-extractor-filters [400]</td>
<td>Replication service extractor filters</td>
</tr>
<tr>
<td>repl-svc-parallelization-type [400], repl-svc-parallelization-type [400]</td>
<td>repl-svc-parallelization-type [400], svc-parallelization-type [400]</td>
<td>Method for implementing parallel apply</td>
</tr>
<tr>
<td>repl-svc-remote-filters [400], repl-svc-remote-filters [400]</td>
<td>repl-svc-remote-filters [400], svc-remote-filters [400]</td>
<td>Replication service remote download filters</td>
</tr>
<tr>
<td>repl-svc-reposition-on-source-id-change [400], repl-svc-reposition-on-source-id-change [400]</td>
<td>repl-svc-reposition-on-source-id-change [400], svc-reposition-on-source-id-change [400]</td>
<td>The master will come ONLINE from the current position if the stored source_id does not match the value in the static properties</td>
</tr>
<tr>
<td>repl-svc-shard-default-db [400], repl-svc-shard-default-db [400]</td>
<td>repl-svc-shard-default-db [400], svc-shard-default-db [400]</td>
<td>Mode for setting the shard ID from the default db</td>
</tr>
<tr>
<td>repl-svc-thl-filters [401], repl-svc-thl-filters [401]</td>
<td>repl-svc-thl-filters [401], svc-thl-filters [401]</td>
<td>Replication service THL filters</td>
</tr>
<tr>
<td>repl-target-dataservice [401], repl-dataservice [401]</td>
<td>repl-target-dataservice [401], target-dataservice [401]</td>
<td>Dataservice to use to determine the value of host configuration</td>
</tr>
<tr>
<td>repl-template-file-help [401]</td>
<td>repl-template-file-help [401]</td>
<td>Display the keys that may be used in configuration template files</td>
</tr>
</tbody>
</table>
### The tpm Deployment Command

<table>
<thead>
<tr>
<th>CmdLine Option</th>
<th>INI File Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--template-search-path [401]</td>
<td>template-search-path [401]</td>
<td>Adds a new template search path for configuration file generation</td>
</tr>
<tr>
<td>--thl-interface [402], repl-thl-interface [402]</td>
<td>repl-thl-interface [402], thl-interface [402]</td>
<td>Listen interface to use for THL operations</td>
</tr>
<tr>
<td>--thl-port [403], repl-thl-port [403]</td>
<td>repl-thl-port [403], thl-port [403]</td>
<td>Port to use for THL Operations</td>
</tr>
<tr>
<td>--thl-protocol [403], repl-thl-protocol [403]</td>
<td>repl-thl-protocol [403], thl-protocol [403]</td>
<td>Protocol to use for THL communication with this service</td>
</tr>
<tr>
<td>--topology [403], dataservice-topology [403]</td>
<td>dataservice-topology [403], topology [403]</td>
<td>Replication topology for the dataservice Valid values are star,cluster-slave,master-slave,fan-in,clustered,cluster-alias,all-masters,direct</td>
</tr>
<tr>
<td>--track-schema-changes [403]</td>
<td>track-schema-changes [403]</td>
<td>This will enable filters that track DDL statements and write the resulting change to files on slave hosts. The feature is intended for use in some batch deployments.</td>
</tr>
<tr>
<td>--user [403]</td>
<td>user [403]</td>
<td>System User</td>
</tr>
<tr>
<td>--vertica-dbname [404], repl-vertica-dbname [404]</td>
<td>repl-vertica-dbname [404], vertica-dbname [404]</td>
<td>Name of the database to replicate into</td>
</tr>
<tr>
<td>--witnesses [404], dataservice-witnesses [404]</td>
<td>dataservice-witnesses [404], witnesses [404]</td>
<td>Witness hosts for the dataservice</td>
</tr>
</tbody>
</table>

### 9.8.1. A tpm Options

#### --allow-bidi-unsafe

<table>
<thead>
<tr>
<th>Option</th>
<th>allow-bidi-unsafe [363]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>repl-allow-bidi-unsafe [363]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>allow-bidi-unsafe [363], repl-allow-bidi-unsafe [363]</td>
</tr>
<tr>
<td>Description</td>
<td>Allow unsafe SQL from remote service</td>
</tr>
<tr>
<td>Value Type</td>
<td>boolean</td>
</tr>
<tr>
<td>Valid Values</td>
<td>false</td>
</tr>
</tbody>
</table>

#### --api

<table>
<thead>
<tr>
<th>Option</th>
<th>api [363]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>repl-api [363]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>api [363], repl-api [363]</td>
</tr>
<tr>
<td>Description</td>
<td>Enable the replication API</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>
### The tpm Deployment Command

<table>
<thead>
<tr>
<th>Alias</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>--repl-api-host</strong></td>
<td>Hostname that the replication API should listen on</td>
<td>string</td>
</tr>
<tr>
<td><strong>--api-password</strong></td>
<td>HTTP basic auth password for the replication API</td>
<td>string</td>
</tr>
<tr>
<td><strong>--api-port</strong></td>
<td>Port that the replication API should bind to</td>
<td>string</td>
</tr>
<tr>
<td><strong>--api-user</strong></td>
<td>HTTP basic auth username for the replication API</td>
<td>string</td>
</tr>
<tr>
<td><strong>--application-password</strong></td>
<td>Database password for the connector</td>
<td>string</td>
</tr>
<tr>
<td><strong>--application-port</strong></td>
<td>Port for the connector to listen on</td>
<td>string</td>
</tr>
<tr>
<td><strong>--application-readonly-port</strong></td>
<td>Port for the connector to listen for read-only connections on</td>
<td>string</td>
</tr>
</tbody>
</table>

### Config File Options

<table>
<thead>
<tr>
<th>Alias</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>api-host</strong></td>
<td>Hostname that the replication API should listen on</td>
<td>string</td>
</tr>
<tr>
<td><strong>repl-api-host</strong></td>
<td>HTTP basic auth password for the replication API</td>
<td>string</td>
</tr>
<tr>
<td><strong>api-password</strong></td>
<td>Port that the replication API should bind to</td>
<td>string</td>
</tr>
<tr>
<td><strong>api-user</strong></td>
<td>HTTP basic auth username for the replication API</td>
<td>string</td>
</tr>
<tr>
<td><strong>application-password</strong></td>
<td>Database password for the connector</td>
<td>string</td>
</tr>
<tr>
<td><strong>application-port</strong></td>
<td>Port for the connector to listen on</td>
<td>string</td>
</tr>
<tr>
<td><strong>application-readonly-port</strong></td>
<td>Port for the connector to listen for read-only connections on</td>
<td>string</td>
</tr>
</tbody>
</table>

### Description

- **--api-password**
  - Description: HTTP basic auth password for the replication API
  - Value Type: string

- **--api-port**
  - Description: Port that the replication API should bind to
  - Value Type: string

- **--api-user**
  - Description: HTTP basic auth username for the replication API
  - Value Type: string

- **--application-password**
  - Description: Database password for the connector
  - Value Type: string

- **--application-port**
  - Description: Port for the connector to listen on
  - Value Type: string

- **--application-readonly-port**
  - Description: Port for the connector to listen for read-only connections on
  - Value Type: string
### --application-user

<table>
<thead>
<tr>
<th>Option</th>
<th>--application-user [365]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--connector-user [365]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>application-user [365], connector-user [365]</td>
</tr>
<tr>
<td>Description</td>
<td>Database username for the connector</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

### --auto-enable

<table>
<thead>
<tr>
<th>Option</th>
<th>--auto-enable [365]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-auto-enable [365]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>auto-enable [365], repl-auto-enable [365]</td>
</tr>
<tr>
<td>Description</td>
<td>Auto-enable services after start-up</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

### --auto-recovery-delay-interval

<table>
<thead>
<tr>
<th>Option</th>
<th>--auto-recovery-delay-interval [365]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-auto-recovery-delay-interval [365]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>auto-recovery-delay-interval [365], repl-auto-recovery-delay-interval [365]</td>
</tr>
<tr>
<td>Description</td>
<td>Delay between going OFFLINE and attempting to go ONLINE</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
<tr>
<td>Valid Values</td>
<td>5</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Option</th>
<th>--auto-recovery-max-attempts [365]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-auto-recovery-max-attempts [365]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>auto-recovery-max-attempts [365], repl-auto-recovery-max-attempts [365]</td>
</tr>
<tr>
<td>Description</td>
<td>Maximum number of attempts at automatic recovery</td>
</tr>
<tr>
<td>Value Type</td>
<td>numeric</td>
</tr>
<tr>
<td>Valid Values</td>
<td>0</td>
</tr>
</tbody>
</table>

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 [212] 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

<table>
<thead>
<tr>
<th>Option</th>
<th>--auto-recovery-reset-interval [365]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-auto-recovery-reset-interval [365]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>auto-recovery-reset-interval [365], repl-auto-recovery-reset-interval [365]</td>
</tr>
<tr>
<td>Description</td>
<td>Delay before autorecovery is deemed to have succeeded</td>
</tr>
<tr>
<td>Value Type</td>
<td>numeric</td>
</tr>
<tr>
<td>Valid Values</td>
<td>5</td>
</tr>
</tbody>
</table>

The time in ONLINE [212] 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. Btpm Options

#### --backup-directory

<table>
<thead>
<tr>
<th>Option</th>
<th>--backup-directory [366]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-backup-directory [366]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>backup-directory [366], repl-backup-directory [366]</td>
</tr>
<tr>
<td>Description</td>
<td>Permanent backup storage directory</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
<tr>
<td>Default</td>
<td>(home directory)/backups</td>
</tr>
<tr>
<td>Valid Values</td>
<td>(home directory)/backups</td>
</tr>
</tbody>
</table>

#### --backup-dump-directory

<table>
<thead>
<tr>
<th>Option</th>
<th>--backup-dump-directory [366]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-backup-dump-directory [366]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>backup-dump-directory [366], repl-backup-dump-directory [366]</td>
</tr>
<tr>
<td>Description</td>
<td>Backup temporary dump directory</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### --backup-method

<table>
<thead>
<tr>
<th>Option</th>
<th>--backup-method [366]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-backup-method [366]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>backup-method [366], repl-backup-method [366]</td>
</tr>
<tr>
<td>Description</td>
<td>Database backup method</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
<tr>
<td>Valid Values</td>
<td>ebs-snapshot</td>
</tr>
<tr>
<td></td>
<td>file-copy-snapshot</td>
</tr>
<tr>
<td></td>
<td>mysqldump Use mysqldump</td>
</tr>
<tr>
<td></td>
<td>none</td>
</tr>
<tr>
<td></td>
<td>script Use a custom script</td>
</tr>
<tr>
<td></td>
<td>xtrabackup Use Percona XtraBackup</td>
</tr>
<tr>
<td></td>
<td>xtrabackup-full Use Percona XtraBackup Full</td>
</tr>
<tr>
<td></td>
<td>xtrabackup-incremental Use Percona XtraBackup Incremental</td>
</tr>
</tbody>
</table>

#### --backup-online

<table>
<thead>
<tr>
<th>Option</th>
<th>--backup-online [366]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-backup-online [366]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>backup-online [366], repl-backup-online [366]</td>
</tr>
<tr>
<td>Description</td>
<td>Does the backup script support backing up a datasource while it is ONLINE</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### --backup-retention

<table>
<thead>
<tr>
<th>Option</th>
<th>--backup-retention [366]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-backup-retention [366]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>backup-retention [366], repl-backup-retention [366]</td>
</tr>
<tr>
<td>Description</td>
<td>Number of backups to retain</td>
</tr>
<tr>
<td>Value Type</td>
<td>numeric</td>
</tr>
</tbody>
</table>
### The tpm Deployment Command

#### --backup-script

<table>
<thead>
<tr>
<th>Option</th>
<th>--backup-script [367]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-backup-script [367]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>backup-script [367], repl-backup-script [367]</td>
</tr>
<tr>
<td>Description</td>
<td>What is the path to the backup script</td>
</tr>
<tr>
<td>Value Type</td>
<td>filename</td>
</tr>
</tbody>
</table>

#### --batch-enabled

<table>
<thead>
<tr>
<th>Option</th>
<th>--batch-enabled [367]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>batch-enabled [367]</td>
</tr>
<tr>
<td>Description</td>
<td>Should the replicator service use a batch applier</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### --batch-load-language

<table>
<thead>
<tr>
<th>Option</th>
<th>--batch-load-language [367]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>batch-load-language [367]</td>
</tr>
<tr>
<td>Description</td>
<td>Which script language to use for batch loading</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
<tr>
<td>Valid Values</td>
<td>js</td>
</tr>
<tr>
<td></td>
<td>sql</td>
</tr>
</tbody>
</table>

#### --batch-load-template

<table>
<thead>
<tr>
<th>Option</th>
<th>--batch-load-template [367]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>batch-load-template [367]</td>
</tr>
<tr>
<td>Description</td>
<td>Value for the loadBatchTemplate property</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### --buffer-size

<table>
<thead>
<tr>
<th>Option</th>
<th>--buffer-size [367]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-buffer-size [367], --repl-svc-applier-buffer-size [367], --repl-svc-buffer-size [367]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>buffer-size [367], repl-buffer-size [367], repl-svc-applier-buffer-size [367], repl-svc-buffer-size [367]</td>
</tr>
<tr>
<td>Description</td>
<td>Replicator queue size between stages (min 1)</td>
</tr>
<tr>
<td>Value Type</td>
<td>numeric</td>
</tr>
</tbody>
</table>

#### 9.8.3. C tpm Options

#### --channels

<table>
<thead>
<tr>
<th>Option</th>
<th>--channels [367]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-channels [367]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>channels [367], repl-channels [367]</td>
</tr>
<tr>
<td>Description</td>
<td>Number of replication channels to use for services</td>
</tr>
<tr>
<td>Value Type</td>
<td>numeric</td>
</tr>
</tbody>
</table>

#### --cluster-slave-auto-recovery-delay-interval

<table>
<thead>
<tr>
<th>Option</th>
<th>--cluster-slave-auto-recovery-delay-interval [367]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--cluster-slave-repl-auto-recovery-delay-interval [367]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>cluster-slave-auto-recovery-delay-interval [367], cluster-slave-repl-auto-recovery-delay-interval [367]</td>
</tr>
</tbody>
</table>

---

367
### 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 | [368] |
| Aliases | --cluster-slave-repl-auto-recovery-max-attempts | [368] |
| Config File Options | cluster-slave-auto-recovery-max-attempts, cluster-slave-repl-auto-recovery-max-attempts | [368] |
| 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 | [368] |
| Aliases | --cluster-slave-repl-auto-recovery-reset-interval | [368] |
| Config File Options | cluster-slave-auto-recovery-reset-interval, cluster-slave-repl-auto-recovery-reset-interval | [368] |
| Description | Default value for --auto-recovery-reset-interval when --topology=cluster-slave |

| Value Type | string |

#### --composite-datasources

| Option | --composite-datasources | [368] |
| Aliases | --dataservice-composite-datasources | [368] |
| Config File Options | composite-datasources, dataservice-composite-datasources | [368] |
| Description | Data services that should be added to this composite data service |

| Value Type | string |

#### --config-file-help

| Option | --config-file-help | [368] |
| Config File Options | config-file-help | [368] |
| Description | Display help information for content of the config file |

| Value Type | string |

#### --conn-java-enable-concurrent-gc

| Option | --conn-java-enable-concurrent-gc | [368] |
| Config File Options | conn-java-enable-concurrent-gc | [368] |
| Description | Connector Java uses concurrent garbage collection |

| Value Type | string |

#### --conn-java-mem-size

| Option | --conn-java-mem-size | [368] |
| Config File Options | conn-java-mem-size | [368] |
| Description | Connector Java heap memory size used to buffer data between clients and databases |
| Value Type | numeric |

| Valid Values | 256 |

The Connector allocates memory for each concurrent client connection, and may use up to the size of the configured MySQL `max_allowed_packet`. With multiple connections, the heap size should be configured to at least the combination of the number of concurrent connections multiplied by the maximum packet size.

#### --conn-round-robin-include-master

| Option | --conn-round-robin-include-master | [368] |
The `tpm Deployment Command`

<table>
<thead>
<tr>
<th>Config File Options</th>
<th>conn-round-robin-include-master [368]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Should the Connector include the master in round-robin load balancing</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

**--connector-affinity**

<table>
<thead>
<tr>
<th>Option</th>
<th>--connector-affinity [369]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>connector-affinity [369]</td>
</tr>
<tr>
<td>Description</td>
<td>The default affinity for all connections</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

**--connector-autoreconnect**

<table>
<thead>
<tr>
<th>Option</th>
<th>--connector-autoreconnect [369]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>connector-autoreconnect [369]</td>
</tr>
<tr>
<td>Description</td>
<td>Enable auto-reconnect in the connector</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

**--connector-autoreconnect-killed-connections**

<table>
<thead>
<tr>
<th>Option</th>
<th>--connector-autoreconnect-killed-connections [369]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>connector-autoreconnect-killed-connections [369]</td>
</tr>
<tr>
<td>Description</td>
<td>Enable autoreconnect for connections killed within the connector</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
<tr>
<td>Valid Values</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td>true</td>
</tr>
</tbody>
</table>

By default, the connector operates as follows:

- Reconnect closed connections
- Retry autocommitted reads

The behavior can be modified by using the `--connector-autoreconnect-killed-connections [369]`. Setting to `false` disables the reconnection or retry of a connection outside of a planned switch or automatic failover. The default is `true`, reconnecting and retrying all connections.

**--connector-bridge-mode**

<table>
<thead>
<tr>
<th>Option</th>
<th>--connector-bridge-mode [369]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--enable-connector-bridge-mode [369]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>connector-bridge-mode [369], enable-connector-bridge-mode [369]</td>
</tr>
<tr>
<td>Description</td>
<td>Enable the Tungsten Connector bridge mode</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

**--connector-default-schema**

<table>
<thead>
<tr>
<th>Option</th>
<th>--connector-default-schema [369]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--connector-forced-schema [369]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>connector-default-schema [369], connector-forced-schema [369]</td>
</tr>
<tr>
<td>Description</td>
<td>Default schema for the connector to use</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

**--connector-delete-user-map**

<table>
<thead>
<tr>
<th>Option</th>
<th>--connector-delete-user-map [369]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>connector-delete-user-map [369]</td>
</tr>
</tbody>
</table>

369
### The tpm Deployment Command

#### Description
Overwrite an existing user.map file

#### Value Type
string

---

#### Option
--connector-disable-connection-warnings [370]

#### Config File Options
can be listed here

#### Description
Hide Connector warnings in log files

#### Value Type
string

---

#### Option
--connector-disconnect-timeout [370]

#### Config File Options
can be listed here

#### Description
Time (in seconds) to wait for active connection to disconnect before forcing them closed [default: 5]

#### Value Type
boolean

---

#### Option
--connector-drop-after-max-connections [370]

#### Config File Options
can be listed here

#### Description
Instantly drop connections that arrive after --connector-max-connections has been reached

#### Value Type
boolean

---

#### Option
--connector-keepalive-timeout [370]

#### Config File Options
can be listed here

#### Description
The Connector keep-alive SELECT statement is submitted if the time since the last activity reaches this timeout value.

#### Value Type
numeric

---

#### Option
--connector-listen-interface [370]

#### Config File Options
can be listed here

#### Description
Listen interface to use for the connector

#### Value Type
string

---

#### Option
--connector-max-connections [370]

#### Config File Options
can be listed here

#### Description
The maximum number of connections the connector should allow at any time

#### Value Type
numeric

---

#### Option
--connector-max-slave-latency [370]

#### Aliases
--connector-max-applied-latency [370]

#### Config File Options
can be listed here

#### Description
The maximum applied latency for slave connections

#### Value Type
string

---

#### Option
--connector-readonly [370]
### The tpm Deployment Command

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>--enable-connector-readonly</strong></td>
<td>Enable the Tungsten Connector read-only mode</td>
<td>string</td>
</tr>
<tr>
<td><strong>--connector-ro-addresses</strong></td>
<td>Connector addresses that should receive a r/o connection</td>
<td>string</td>
</tr>
<tr>
<td><strong>--connector-rw-addresses</strong></td>
<td>Connector addresses that should receive a r/w connection</td>
<td>string</td>
</tr>
<tr>
<td><strong>--connector-rwsplitting</strong></td>
<td>Enable DirectReads R/W splitting in the connector</td>
<td>string</td>
</tr>
<tr>
<td><strong>--connector-smartscale</strong></td>
<td>Enable SmartScale R/W splitting in the connector</td>
<td>string</td>
</tr>
<tr>
<td><strong>--connector-smartscale-sessionid</strong></td>
<td>The default session ID to use with smart scale</td>
<td>string</td>
</tr>
<tr>
<td><strong>--connectors</strong></td>
<td>Hostnames for the dataservice connectors</td>
<td>string</td>
</tr>
<tr>
<td><strong>--consistency-policy</strong></td>
<td>Should the replicator stop or warn if a consistency check fails?</td>
<td>string</td>
</tr>
</tbody>
</table>

**Aliases**

- **--enable-connector-readonly**
- **--connector-ro-addresses**
- **--connector-rw-addresses**
- **--connector-rwsplitting**
- **--connector-smartscale**
- **--connector-smartscale-sessionid**
- **--connectors**
- **--consistency-policy**
### 9.8.4. **tpm Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Config File Options</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>--dataservice-name</strong></td>
<td>dataservice-name  [372]</td>
<td>Limit the command to the hosts in this dataservice. Multiple data services may be specified by providing a comma separated list.</td>
<td>string</td>
</tr>
<tr>
<td><strong>--dataservice-relay-enabled</strong></td>
<td>dataservice-relay-enabled [372]</td>
<td>Make this dataservice the slave of another.</td>
<td>string</td>
</tr>
<tr>
<td><strong>--dataservice-schema</strong></td>
<td>dataservice-schema  [372]</td>
<td>The db schema to hold dataservice details.</td>
<td>string</td>
</tr>
<tr>
<td><strong>--dataservice-thl-port</strong></td>
<td>dataservice-thl-port  [372]</td>
<td>Port to use for THL operations.</td>
<td>string</td>
</tr>
<tr>
<td><strong>--dataservice-use-relative-latency</strong></td>
<td>dataservice-use-relative-latency [372]</td>
<td>Enable the cluster to operate on relative latency.</td>
<td>string</td>
</tr>
<tr>
<td><strong>--dataservice-vip-enabled</strong></td>
<td>dataservice-vip-enabled [372]</td>
<td>Is VIP management enabled?</td>
<td>string</td>
</tr>
<tr>
<td><strong>--dataservice-vip-ipaddress</strong></td>
<td>dataservice-vip-ipaddress [372]</td>
<td>VIP IP address.</td>
<td>string</td>
</tr>
<tr>
<td><strong>--dataservice-vip-netmask</strong></td>
<td>dataservice-vip-netmask [372]</td>
<td>VIP IP netmask.</td>
<td>string</td>
</tr>
</tbody>
</table>
### The `tpm` Deployment Command

<table>
<thead>
<tr>
<th>Option</th>
<th>Config File Options</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>--dataservice-vip-netmask</td>
<td>dataservice-vip-netmask</td>
<td>VIP netmask</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--datasource-boot-script</td>
<td>datasource-boot-script</td>
<td>Database start script</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td>repl-datasource-boot-script</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--datasource-enable-ssl</td>
<td>datasource-enable-ssl</td>
<td>Enable SSL connection to DBMS server</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td>repl-datasource-enable-ssl</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--datasource-log-directory</td>
<td>datasource-log-directory</td>
<td>Master log directory</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td>repl-datasource-log-directory</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--datasource-log-pattern</td>
<td>datasource-log-pattern</td>
<td>Master log filename pattern</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td>repl-datasource-log-pattern</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--datasource-mysql-conf</td>
<td>datasource-mysql-conf</td>
<td>MySQL config file</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td>repl-datasource-mysql-conf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--datasource-mysql-data-directory</td>
<td>datasource-mysql-data-directory</td>
<td>MySQL data directory</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td>repl-datasource-mysql-data-directory</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--datasource-mysql-ibdata-directory</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## The `tpm` Deployment Command

<table>
<thead>
<tr>
<th>Option</th>
<th>--datasource-mysql-ibdata-directory [373]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-datasource-mysql-ibdata-directory [373]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>datasource-mysql-ibdata-directory [373], repl-datasource-mysql-ibdata-directory [373]</td>
</tr>
<tr>
<td>Description</td>
<td>MySQL InnoDB data directory</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>--datasource-mysql-iblog-directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option</td>
</tr>
<tr>
<td>Aliases</td>
</tr>
<tr>
<td>Config File Options</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td>Value Type</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>--datasource-mysql-ssl-ca</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option</td>
</tr>
<tr>
<td>Aliases</td>
</tr>
<tr>
<td>Config File Options</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td>Value Type</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>--datasource-mysql-ssl-cert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option</td>
</tr>
<tr>
<td>Aliases</td>
</tr>
<tr>
<td>Config File Options</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td>Value Type</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>--datasource-mysql-ssl-key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option</td>
</tr>
<tr>
<td>Aliases</td>
</tr>
<tr>
<td>Config File Options</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td>Value Type</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>--datasource-oracle-scan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option</td>
</tr>
<tr>
<td>Aliases</td>
</tr>
<tr>
<td>Config File Options</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td>Value Type</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>--datasource-oracle-service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option</td>
</tr>
<tr>
<td>Aliases</td>
</tr>
<tr>
<td>Config File Options</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td>Value Type</td>
</tr>
</tbody>
</table>
The `tpm` Deployment Command

### `--datasource-oracle-service-group`

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--datasource-oracle-service-group</code></td>
<td></td>
<td>Oracle service operating system group</td>
</tr>
</tbody>
</table>

The name of the operating system group that Oracle users should be a member of. If this value cannot be determined or is not set properly, the Tungsten Clustering will be unable to access the necessary Oracle tools.

### `--datasource-pg-archive`

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--datasource-pg-archive</code></td>
<td></td>
<td>PostgreSQL archive location</td>
</tr>
</tbody>
</table>

### `--datasource-pg-conf`

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--datasource-pg-conf</code></td>
<td></td>
<td>Location of postgresql.conf</td>
</tr>
</tbody>
</table>

### `--datasource-pg-home`

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--datasource-pg-home</code></td>
<td></td>
<td>PostgreSQL data directory</td>
</tr>
</tbody>
</table>

### `--datasource-pg-root`

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--datasource-pg-root</code></td>
<td></td>
<td>Root directory for postgresql installation</td>
</tr>
</tbody>
</table>

### `--datasource-systemctl-service`

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--datasource-systemctl-service</code></td>
<td></td>
<td>Database systemctl script</td>
</tr>
</tbody>
</table>

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 Clustering 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`

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
<th>Description</th>
</tr>
</thead>
</table>
The tpm Deployment Command

<table>
<thead>
<tr>
<th>Option</th>
<th>--datasource-type [375]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-datasource-type [375]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>datasource-type [375], repl-datasource-type [375]</td>
</tr>
<tr>
<td>Description</td>
<td>Database type</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
<tr>
<td>Default</td>
<td>mysql</td>
</tr>
<tr>
<td>Valid Values</td>
<td>file</td>
</tr>
<tr>
<td></td>
<td>hdfs</td>
</tr>
<tr>
<td></td>
<td>kafka</td>
</tr>
<tr>
<td></td>
<td>mongodb</td>
</tr>
<tr>
<td></td>
<td>mysql</td>
</tr>
<tr>
<td></td>
<td>oracle</td>
</tr>
<tr>
<td></td>
<td>postgres</td>
</tr>
<tr>
<td></td>
<td>vertica</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>--delete [376]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>delete [376]</td>
</tr>
<tr>
<td>Description</td>
<td>Delete the named data service from the configuration Data Service options:</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>--direct-datasource-log-directory [376]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-direct-datasource-log-directory [376]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>direct-datasource-log-directory [376], repl-direct-datasource-log-directory [376]</td>
</tr>
<tr>
<td>Description</td>
<td>Master log directory</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>--direct-datasource-log-pattern [376]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-direct-datasource-log-pattern [376]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>direct-datasource-log-pattern [376], repl-direct-datasource-log-pattern [376]</td>
</tr>
<tr>
<td>Description</td>
<td>Master log filename pattern</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>--direct-datasource-oracle-scan [376]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-direct-datasource-oracle-scan [376]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>direct-datasource-oracle-scan [376], repl-direct-datasource-oracle-scan [376]</td>
</tr>
<tr>
<td>Description</td>
<td>Oracle SCAN</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>--direct-datasource-oracle-service [376]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-direct-datasource-oracle-service [376]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>direct-datasource-oracle-service [376], repl-direct-datasource-oracle-service [376]</td>
</tr>
<tr>
<td>Description</td>
<td>Oracle Service</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td><strong>string</strong></td>
</tr>
<tr>
<td>----------------</td>
<td>------------</td>
</tr>
</tbody>
</table>

**--direct-datasource-oracle-sid**

<table>
<thead>
<tr>
<th><strong>Option</strong></th>
<th>--direct-datasource-oracle-sid [377]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Aliases</strong></td>
<td>--repl-direct-datasource-oracle-sid [377]</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>direct-datasource-oracle-sid [377], repl-direct-datasource-oracle-sid [377]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Oracle SID</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

**--direct-datasource-type**

<table>
<thead>
<tr>
<th><strong>Option</strong></th>
<th>--direct-datasource-type [377]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Aliases</strong></td>
<td>--repl-direct-datasource-type [377]</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>direct-datasource-type [377], repl-direct-datasource-type [377]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Database type</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
<tr>
<td><strong>Valid Values</strong></td>
<td>mysql, hdfs, mongodb, mysql, mysql, oracle, vertica</td>
</tr>
<tr>
<td><strong>Default</strong></td>
<td>mysql</td>
</tr>
<tr>
<td><strong>Valid Values</strong></td>
<td>file, HDFS [Hadoop], MySQL, Oracle, Vertica</td>
</tr>
</tbody>
</table>

**--direct-replication-host**

<table>
<thead>
<tr>
<th><strong>Option</strong></th>
<th>--direct-replication-host [377]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Aliases</strong></td>
<td>--direct-datasource-host [377], --repl-direct-datasource-host [377]</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>direct-datasource-host [377], direct-replication-host [377], repl-direct-datasource-host [377]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Database server hostname</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

**--direct-replication-password**

<table>
<thead>
<tr>
<th><strong>Option</strong></th>
<th>--direct-replication-password [377]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Aliases</strong></td>
<td>--direct-datasource-password [377], --repl-direct-datasource-password [377]</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>direct-datasource-password [377], direct-replication-password [377], repl-direct-datasource-password [377]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Password for datasource connection</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

**--direct-replication-port**

<table>
<thead>
<tr>
<th><strong>Option</strong></th>
<th>--direct-replication-port [377]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Aliases</strong></td>
<td>--direct-datasource-port [377], --repl-direct-datasource-port [377]</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>direct-datasource-port [377], direct-replication-port [377], repl-direct-datasource-port [377]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Database server port</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

**--direct-replication-user**

| **Option** | --direct-replication-user [377] |
## 9.8.5. `tpm` Options

### `--enable-active-witnesses`

<table>
<thead>
<tr>
<th>Option</th>
<th><code>--enable-active-witnesses [378]</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td><code>--active-witnesses [378]</code></td>
</tr>
<tr>
<td>Config File Options</td>
<td><code>active-witnesses [378], enable-active-witnesses [378]</code></td>
</tr>
<tr>
<td>Description</td>
<td>Enable active witness hosts</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

### `--enable-batch-master`

<table>
<thead>
<tr>
<th>Option</th>
<th><code>--enable-batch-master [378]</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td><code>--active-witnesses [378]</code></td>
</tr>
<tr>
<td>Config File Options</td>
<td><code>active-witnesses [378], enable-active-witnesses [378]</code></td>
</tr>
<tr>
<td>Description</td>
<td>Enable active witness hosts</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>
The tpdm Deployment Command

<table>
<thead>
<tr>
<th>Option</th>
<th>--enable-batch-master [378]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>enable-batch-master [378]</td>
</tr>
<tr>
<td>Description</td>
<td>Enable batch operation for the master</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

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 Master
  - `mysql-use-bytes-for-string` 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` stages), 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 Slave
  - `mysql-use-bytes-for-string` is set to true.
  - `pkey` filter is enabled (in the `q-to-dbms` stage).

<table>
<thead>
<tr>
<th>Option</th>
<th>--enable-batch-service [379]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>enable-batch-service [379]</td>
</tr>
<tr>
<td>Description</td>
<td>Enables batch mode for a service</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
<tr>
<td>Valid Values</td>
<td>false</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>--enable-batch-slave [379]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>enable-batch-slave [379]</td>
</tr>
<tr>
<td>Description</td>
<td>Enable batch operation for the slave</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>--enable-connector-client-ssl [379]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--connector-client-ssl [379]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>connector-client-ssl [379], enable-connector-client-ssl [379]</td>
</tr>
<tr>
<td>Description</td>
<td>Enable SSL encryption of traffic from the client to the connector</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>--enable-connector-server-ssl [379]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--connector-server-ssl [379]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>connector-server-ssl [379], enable-connector-server-ssl [379]</td>
</tr>
<tr>
<td>Description</td>
<td>Enable SSL encryption of traffic from the connector to the database</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>--enable-connector-ssl</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--connector-ssl</td>
</tr>
<tr>
<td>Config File Options</td>
<td>connector-ssl [379], enable-connector-ssl [379]</td>
</tr>
<tr>
<td>Description</td>
<td>Enable SSL encryption of traffic from the client to the database</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>
## The tpm Deployment Command

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>--enable-connector-ssl</strong></td>
<td>Enable SSL encryption of connector traffic to the database</td>
</tr>
<tr>
<td><strong>--enable-heterogeneous-master</strong></td>
<td>Enable heterogeneous operation for the master</td>
</tr>
<tr>
<td><strong>--enable-heterogeneous-service</strong></td>
<td>Enable heterogeneous operation</td>
</tr>
<tr>
<td><strong>--enable-heterogeneous-slave</strong></td>
<td>Enable heterogeneous operation for the slave</td>
</tr>
<tr>
<td><strong>--enable-jgroups-ssl</strong></td>
<td>Enable SSL encryption of JGroups communication on this host</td>
</tr>
<tr>
<td><strong>--enable-rmi-authentication</strong></td>
<td>Enable RMI authentication</td>
</tr>
</tbody>
</table>

### Option Descriptions

#### --enable-connector-ssl
- **Description**: Enable SSL encryption of connector traffic to the database.
- **Value Type**: string

#### --enable-heterogeneous-master
- **Description**: Enable heterogeneous operation for the master.
- **Value Type**: string

#### --enable-heterogeneous-service
- **Description**: Enable heterogeneous operation.
- **Value Type**: string

#### --enable-heterogeneous-slave
- **Description**: Enable heterogeneous operation for the slave.
- **Value Type**: string

#### --enable-jgroups-ssl
- **Description**: Enable SSL encryption of JGroups communication on this host.
- **Value Type**: string

#### --enable-rmi-authentication
- **Description**: Enable RMI authentication.

### Additional Notes

- **On a Master**
  - `--mysql-use-bytes-for-string` is set to false.
  - The `colnames` filter is enabled (in the binlog-to-q stage to add column names to the THL information).
  - The `pkey` filter is enabled (in the binlog-to-q and q-to-dbms stage), with the `addPkeyToInserts` and `addColumnsToDeletes` filter options set to false.
  - The `enumtoString` filter is enabled (in the q-to-thl stage), to translate ENUM values to their string equivalents.
  - The `settoString` filter is enabled (in the q-to-thl stage), to translate SET values to their string equivalents.

- **On a Slave**
  - `--mysql-use-bytes-for-string` is set to true.
  - The `pkey` filter is enabled (q-to-dbms stage).
### The tpm Deployment Command

<table>
<thead>
<tr>
<th>Aliases</th>
<th>--rmi-authentication [380]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>enable-rmi-authentication [380], rmi-authentication [380]</td>
</tr>
<tr>
<td>Description</td>
<td>Enable RMI authentication for the services running on this host</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

--enable-rmi-ssl

<table>
<thead>
<tr>
<th>Option</th>
<th>--enable-rmi-ssl [381]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--rmi-ssl [381]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>enable-rmi-ssl [381], rmi-ssl [381]</td>
</tr>
<tr>
<td>Description</td>
<td>Enable SSL encryption of RMI communication on this host</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

--enable-slave-thl-listener

<table>
<thead>
<tr>
<th>Option</th>
<th>--enable-slave-thl-listener [381]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-enable-slave-thl-listener [381]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>enable-slave-thl-listener [381], repl-enable-slave-thl-listener [381]</td>
</tr>
<tr>
<td>Description</td>
<td>Should this service allow THL connections?</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

--enable-sudo-access

<table>
<thead>
<tr>
<th>Option</th>
<th>--enable-sudo-access [381]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--root-command-prefix [381]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>enable-sudo-access [381], root-command-prefix [381]</td>
</tr>
<tr>
<td>Description</td>
<td>Run root commands using sudo</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

--enable-thl-ssl

<table>
<thead>
<tr>
<th>Option</th>
<th>--enable-thl-ssl [381]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-enable-thl-ssl [381], --thl-ssl [381]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>enable-thl-ssl [381], repl-enable-thl-ssl [381], thl-ssl [381]</td>
</tr>
<tr>
<td>Description</td>
<td>Enable SSL encryption of THL communication for this service</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

--executable-prefix

<table>
<thead>
<tr>
<th>Option</th>
<th>--executable-prefix [381]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>executable-prefix [381]</td>
</tr>
<tr>
<td>Description</td>
<td>Adds a prefix to command aliases</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

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 be accessible through a unique alias. For example, if the executable prefix is configured as `east`, then an alias for the installation to `trepc` will be created as `east_trepc`.

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

<table>
<thead>
<tr>
<th>Option</th>
<th>--file-protection-level [381]</th>
</tr>
</thead>
</table>
Config File Options | file-protection-level [381]  
Description | Protection level for Continuent files  
Value Type | string  

--file-protection-umask  
Option | --file-protection-umask [382]  
Config File Options | file-protection-umask [382]  
Description | Protection umask for Continuent files  
Value Type | string  

9.8.7. **H tpm Options**  

--host-name  
Option | --host-name [382]  
Config File Options | host-name [382]  
Description | DNS hostname  
Value Type | string  

--hosts  
Option | --hosts [382]  
Config File Options | hosts [382]  
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 [382]  
Aliases | --dataservice-hub-host [382]  
Config File Options | dataservice-hub-host [382], hub [382]  
Description | What is the hub host for this all-masters dataservice?  
Value Type | string  

--hub-service  
Option | --hub-service [382]  
Aliases | --dataservice-hub-service [382]  
Config File Options | dataservice-hub-service [382], hub-service [382]  
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 [382]  
Config File Options | install [382]  
Description | Install service start scripts  
Value Type | string  

--install-directory  
Option | --install-directory [382]
### 9.8.9. J tpm Options

#### --java-connector-keystore-password

<table>
<thead>
<tr>
<th>Option</th>
<th>--java-connector-keystore-password [383]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>java-connector-keystore-password [383]</td>
</tr>
<tr>
<td>Description</td>
<td>The password for unlocking the tungsten_connector_keystore.jks file in the security directory</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### --java-connector-keystore-path

<table>
<thead>
<tr>
<th>Option</th>
<th>--java-connector-keystore-path [383]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>java-connector-keystore-path [383]</td>
</tr>
<tr>
<td>Description</td>
<td>Local path to the Java Connector Keystore file.</td>
</tr>
<tr>
<td>Value Type</td>
<td>filename</td>
</tr>
</tbody>
</table>

#### --java-connector-truststore-password

<table>
<thead>
<tr>
<th>Option</th>
<th>--java-connector-truststore-password [383]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>java-connector-truststore-password [383]</td>
</tr>
<tr>
<td>Description</td>
<td>The password for unlocking the tungsten_connector_truststore.jks file in the security directory</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### --java-connector-truststore-path

<table>
<thead>
<tr>
<th>Option</th>
<th>--java-connector-truststore-path [383]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>java-connector-truststore-path [383]</td>
</tr>
<tr>
<td>Description</td>
<td>Local path to the Java Connector Truststore file.</td>
</tr>
<tr>
<td>Value Type</td>
<td>filename</td>
</tr>
</tbody>
</table>

#### --java-enable-concurrent-gc

<table>
<thead>
<tr>
<th>Option</th>
<th>--java-enable-concurrent-gc [383]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-java-enable-concurrent-gc [383]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>java-enable-concurrent-gc [383], repl-java-enable-concurrent-gc [383]</td>
</tr>
<tr>
<td>Description</td>
<td>Replicator Java uses concurrent garbage collection</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### --java-external-lib-dir

<table>
<thead>
<tr>
<th>Option</th>
<th>--java-external-lib-dir [383]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-java-external-lib-dir [383]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>java-external-lib-dir [383], repl-java-external-lib-dir [383]</td>
</tr>
<tr>
<td>Description</td>
<td>Directory for 3rd party Jar files required by replicator</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### --java-file-encoding

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.
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>--java-file-encoding</strong></td>
<td>Java platform charset (esp. for heterogeneous replication)</td>
</tr>
<tr>
<td><strong>--repl-java-file-encoding</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td><strong>java-file-encoding</strong>, <strong>repl-java-file-encoding</strong></td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

**--java-jgroups-key**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>--java-jgroups-key</strong></td>
<td>The alias to use for the JGroups TLS key in the keystore.</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td><strong>java-jgroups-key</strong></td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

**--java-jgroups-keystore-path**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>--java-jgroups-keystore-path</strong></td>
<td>Local path to the JGroups Java Keystore file.</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td><strong>java-jgroups-keystore-path</strong></td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td><strong>filename</strong></td>
</tr>
</tbody>
</table>

**--java-jmxremote-access-path**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>--java-jmxremote-access-path</strong></td>
<td>Local path to the Java JMX Remote Access file.</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td><strong>java-jmxremote-access-path</strong></td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td><strong>filename</strong></td>
</tr>
</tbody>
</table>

**--java-keystore-password**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>--java-keystore-password</strong></td>
<td>The password for unlocking the tungsten_keystore.jks file in the security directory</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td><strong>java-keystore-password</strong></td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

**--java-keystore-path**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>--java-keystore-path</strong></td>
<td>Local path to the Java Keystore file.</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td><strong>java-keystore-path</strong></td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>filename</td>
</tr>
</tbody>
</table>

**--java-mem-size**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>--java-mem-size</strong></td>
<td>Replicator Java heap memory size in Mb [min 128]</td>
</tr>
<tr>
<td><strong>Aliases</strong></td>
<td><strong>repl-java-mem-size</strong></td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td><strong>java-mem-size</strong>, <strong>repl-java-mem-size</strong></td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>numeric</td>
</tr>
</tbody>
</table>

**--java-passwordstore-path**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>--java-passwordstore-path</strong></td>
<td>Local path to the Java Password Store file.</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td><strong>java-passwordstore-path</strong></td>
</tr>
</tbody>
</table>
### 9.8.10. L tpm Options

|--log

<table>
<thead>
<tr>
<th>Option</th>
<th>--log [385]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>log [385]</td>
</tr>
<tr>
<td>Description</td>
<td>Write all messages, visible and hidden, to this file. You may specify a filename, 'pid' or 'timestamp'.</td>
</tr>
<tr>
<td>Value Type</td>
<td>numeric</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>log-slave-updates</td>
<td>Should slaves log updates to binlog</td>
</tr>
<tr>
<td><strong>9.8.11. M tpm Options</strong></td>
<td></td>
</tr>
</tbody>
</table>

**--master**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>master</td>
<td>Hostname of the master host within this service</td>
<td>string</td>
</tr>
<tr>
<td>master-preferred-role</td>
<td>Preferred role for master THL when connecting as a slave</td>
<td>string</td>
</tr>
<tr>
<td>master-services</td>
<td>Data service names that should be used on each master</td>
<td>string</td>
</tr>
<tr>
<td>master-thl-host</td>
<td>Master THL Hostname</td>
<td>string</td>
</tr>
<tr>
<td>master-thl-port</td>
<td>Master THL Port</td>
<td>string</td>
</tr>
<tr>
<td>members</td>
<td></td>
<td>string</td>
</tr>
</tbody>
</table>
## Config File Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--dataservice-hosts</td>
<td>Hostnames for the dataservice members</td>
</tr>
<tr>
<td>--members</td>
<td></td>
</tr>
<tr>
<td>--metadata-directory</td>
<td>Replicator metadata directory</td>
</tr>
<tr>
<td>--mgr-api</td>
<td>Enable the Manager API</td>
</tr>
<tr>
<td>--mgr-api-address</td>
<td>Address for the Manager API</td>
</tr>
<tr>
<td>--mgr-api-full-access</td>
<td>Enable all Manager API commands. Only the status command will be enabled without it.</td>
</tr>
<tr>
<td>--mgr-api-port</td>
<td>Port for the Manager API</td>
</tr>
<tr>
<td>--mgr-group-communication-port</td>
<td>Port to use for manager group communication</td>
</tr>
<tr>
<td>--mgr-heap-threshold</td>
<td>Java memory usage (MB) that will force a Manager restart</td>
</tr>
<tr>
<td>--mgr-java-enable-concurrent-gc</td>
<td></td>
</tr>
</tbody>
</table>
### The tpm Deployment Command

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--mgr-java-enable-concurrent-gc</td>
<td>Manager Java uses concurrent garbage collection</td>
</tr>
<tr>
<td>config_file_option</td>
<td>--mgr-java-enable-concurrent-gc</td>
</tr>
<tr>
<td>value_type</td>
<td>string</td>
</tr>
</tbody>
</table>

**--mgr-java-mem-size**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--mgr-java-mem-size</td>
<td>Manager Java heap memory size in Mb (min 128)</td>
</tr>
<tr>
<td>config_file_option</td>
<td>--mgr-java-mem-size</td>
</tr>
<tr>
<td>value_type</td>
<td>numeric</td>
</tr>
</tbody>
</table>

**--mgr-listen-interface**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--mgr-listen-interface</td>
<td>Listen interface to use for the manager</td>
</tr>
<tr>
<td>config_file_option</td>
<td>--mgr-listen-interface</td>
</tr>
<tr>
<td>value_type</td>
<td>string</td>
</tr>
</tbody>
</table>

**--mgr-ping-method**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--mgr-ping-method</td>
<td>Mechanism to use when identifying the liveness of other datasources (ping, echo)</td>
</tr>
<tr>
<td>config_file_option</td>
<td>--mgr-ping-method</td>
</tr>
<tr>
<td>value_type</td>
<td>string</td>
</tr>
</tbody>
</table>

**--mgr-policy-mode**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--mgr-policy-mode</td>
<td>Manager policy mode</td>
</tr>
<tr>
<td>config_file_option</td>
<td>--mgr-policy-mode</td>
</tr>
<tr>
<td>value_type</td>
<td>string</td>
</tr>
<tr>
<td>valid_values</td>
<td>automatic, maintenance, manual</td>
</tr>
</tbody>
</table>

**--mgr-rmi-port**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--mgr-rmi-port</td>
<td>Port to use for the manager RMI server</td>
</tr>
<tr>
<td>config_file_option</td>
<td>--mgr-rmi-port</td>
</tr>
<tr>
<td>value_type</td>
<td>string</td>
</tr>
</tbody>
</table>

**--mgr-rmi-remote-port**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--mgr-rmi-remote-port</td>
<td>Port to use for calling the remote manager RMI server</td>
</tr>
<tr>
<td>config_file_option</td>
<td>--mgr-rmi-remote-port</td>
</tr>
<tr>
<td>value_type</td>
<td>string</td>
</tr>
</tbody>
</table>

**--mgr-ro-slave**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--mgr-ro-slave</td>
<td></td>
</tr>
<tr>
<td>config_file_option</td>
<td>--mgr-ro-slave</td>
</tr>
<tr>
<td>value_type</td>
<td></td>
</tr>
</tbody>
</table>
The tpm Deployment Command

<table>
<thead>
<tr>
<th>Description</th>
<th>Make slaves read-only</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

---

**--mgr-arp-path**

<table>
<thead>
<tr>
<th>Option</th>
<th>--mgr-arp-path [389]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>mgr-arp-path [389]</td>
</tr>
<tr>
<td>Description</td>
<td>Path to the arp binary</td>
</tr>
<tr>
<td>Value Type</td>
<td>filename</td>
</tr>
</tbody>
</table>

---

**--mgr-device**

<table>
<thead>
<tr>
<th>Option</th>
<th>--mgr-device [389]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>mgr-device [389]</td>
</tr>
<tr>
<td>Description</td>
<td>VIP network device</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

---

**--mgr-ifconfig-path**

<table>
<thead>
<tr>
<th>Option</th>
<th>--mgr-ifconfig-path [389]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>mgr-ifconfig-path [389]</td>
</tr>
<tr>
<td>Description</td>
<td>Path to the ifconfig binary</td>
</tr>
<tr>
<td>Value Type</td>
<td>filename</td>
</tr>
</tbody>
</table>

---

**--mgr-wait-for-members**

<table>
<thead>
<tr>
<th>Option</th>
<th>--mgr-wait-for-members [389]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>mgr-wait-for-members [389]</td>
</tr>
<tr>
<td>Description</td>
<td>Wait for all datasources to be available before completing installation</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

---

**--mysql-allow-intensive-checks**

<table>
<thead>
<tr>
<th>Option</th>
<th>--mysql-allow-intensive-checks [389]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>mysql-allow-intensive-checks [389]</td>
</tr>
<tr>
<td>Description</td>
<td>For MySQL installation, enables detailed checks on the supported data types within the MySQL database to confirm compatibility.</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

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-connectorj-path**

<table>
<thead>
<tr>
<th>Option</th>
<th>--mysql-connectorj-path [389]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>mysql-connectorj-path [389]</td>
</tr>
<tr>
<td>Description</td>
<td>Path to MySQL Connector/J</td>
</tr>
<tr>
<td>Value Type</td>
<td>filename</td>
</tr>
</tbody>
</table>

---

**Important**

As of Tungsten Clustering v4.0.0, the MySQL Connector/J prerequisite has been removed. The JDBC interface now uses the Drizzle driver by default.

---

**Note**

tpm reverse will display the parameter --mysql-connectorj-path as long as any mysql-connector-java file remains in /opt/continuent/share/
The tpm Deployment Command

### Warning

Do not use path `/opt/continuent/share/` inside the value for `--mysql-connectorj-path` or tpm will abort with an error.

---

#### Option: `--mysql-driver`

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--mysql-driver</code></td>
<td>string</td>
</tr>
</tbody>
</table>

#### Config File Options

- `mysql-driver`

#### Description

MySQL Driver Vendor

---

#### Option: `--mysql-enable-ansiquotes`

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--mysql-enable-ansiquotes</code></td>
<td>string</td>
</tr>
</tbody>
</table>

#### Aliases

- `repl-mysql-enable-ansiquotes`

#### Config File Options

- `mysql-enable-ansiquotes`
- `repl-mysql-enable-ansiquotes`

#### Description

Enables ANSI_QUOTES mode for incoming events?

---

#### Option: `--mysql-enable-enumtostring`

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--mysql-enable-enumtostring</code></td>
<td>string</td>
</tr>
</tbody>
</table>

#### Aliases

- `repl-mysql-enable-enumtostring`

#### Config File Options

- `mysql-enable-enumtostring`
- `repl-mysql-enable-enumtostring`

#### Description

Enable a filter to convert ENUM values to strings

---

#### Option: `--mysql-enable-noonlykeywords`

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--mysql-enable-noonlykeywords</code></td>
<td>string</td>
</tr>
</tbody>
</table>

#### Aliases

- `repl-mysql-enable-noonlykeywords`

#### Config File Options

- `mysql-enable-noonlykeywords`
- `repl-mysql-enable-noonlykeywords`

#### Description

Enables a filter to translate `DELETE FROM ONLY` to `DELETE FROM` and `UPDATE ONLY` to `UPDATE`.

---

#### Option: `--mysql-enable-settostring`

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--mysql-enable-settostring</code></td>
<td>string</td>
</tr>
</tbody>
</table>

#### Aliases

- `repl-mysql-enable-settostring`

#### Config File Options

- `mysql-enable-settostring`
- `repl-mysql-enable-settostring`

#### Description

Enable a filter to convert SET types to strings

---

#### Option: `--mysql-ro-slave`

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--mysql-ro-slave</code></td>
<td>string</td>
</tr>
</tbody>
</table>

#### Aliases

- `repl-mysql-ro-slave`

#### Config File Options

- `mysql-ro-slave`
- `repl-mysql-ro-slave`

#### Description

Slaves are read-only?

---

#### Option: `--mysql-server-id`

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--mysql-server-id</code></td>
<td>string</td>
</tr>
</tbody>
</table>

#### Aliases

- `repl-mysql-server-id`

---

390
The tpm Deployment Command

| Config File Options | mysql-server-id [390], repl-mysql-server-id [390] |
| Description         | Explicitly set the MySQL server ID |
| Value Type          | string |

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 master.

--mysql-use-bytes-for-string

| Option                        | --mysql-use-bytes-for-string [391] |
| Alias | --repl-mysql-use-bytes-for-string [391] |
| Config File Options | mysql-use-bytes-for-string [391], repl-mysql-use-bytes-for-string [391] |
| Description | Transfer strings as their byte representation? |
| Value Type | string |

9.8.12. N tpm Options

--native-slave-takeover

| Option                        | --native-slave-takeover [391] |
| Alias | --repl-native-slave-takeover [391] |
| Config File Options | native-slave-takeover [391], repl-native-slave-takeover [391] |
| Description | Takeover native replication |
| Value Type | string |

--no-deployment

| Option                        | --no-deployment [391] |
| Config File Options | no-deployment [391] |
| Description | Skip deployment steps that create the install directory |
| Value Type | string |

--no-validation

| Option                        | --no-validation [391] |
| Config File Options | no-validation [391] |
| 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 [391] |
| Config File Options | optimize-row-events [391] |
| Description | Enables or disables optimized row updates |
### Optimized Row Updates

Optimized row updates bundle multiple row-based updates into a single `INSERT` or `UPDATE` statement. This increases the throughput of large batches of row-based updates.

---

#### `--oracle-extractor-method`

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>boolean</td>
<td>Oracle extractor method</td>
</tr>
</tbody>
</table>

#### `--pg-archive-timeout`

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>numeric</td>
<td>Timeout for sending unfilled WAL buffers (data loss window)</td>
</tr>
</tbody>
</table>

#### `--pg-ctl`

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>filename</td>
<td>Path to the pg_ctl script</td>
</tr>
</tbody>
</table>

#### `--pg-method`

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>string</td>
<td>Postgres Replication method</td>
</tr>
</tbody>
</table>

#### `--pg-standby`

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>filename</td>
<td>Path to the pg_standby script</td>
</tr>
</tbody>
</table>

#### `--postgresql-dbname`

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Path to the postgresql-dbname script</td>
</tr>
</tbody>
</table>
### The tpm Deployment Command

<table>
<thead>
<tr>
<th>Description</th>
<th>Name of the database to replicate</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

---

**--postgresql-enable-mysql2pgddl**

<table>
<thead>
<tr>
<th>Option</th>
<th>--postgresql-enable-mysql2pgddl [393]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Aliases</strong></td>
<td>--repl-postgresql-enable-mysql2pgddl [393]</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>postgresql-enable-mysql2pgddl [393], repl-postgresql-enable-mysql2pgddl [393]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Enable MySQL to PostgreSQL DDL dialect converting filter placeholder</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
<tr>
<td><strong>Valid Values</strong></td>
<td>false</td>
</tr>
</tbody>
</table>

---

**--postgresql-slonik**

<table>
<thead>
<tr>
<th>Option</th>
<th>--postgresql-slonik [393]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Aliases</strong></td>
<td>--repl-postgresql-slonik [393]</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>postgresql-slonik [393], repl-postgresql-slonik [393]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Path to the slonik executable</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>filename</td>
</tr>
</tbody>
</table>

---

**--postgresql-tables**

<table>
<thead>
<tr>
<th>Option</th>
<th>--postgresql-tables [393]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Aliases</strong></td>
<td>--repl-postgresql-tables [393]</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>postgresql-tables [393], repl-postgresql-tables [393]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Tables to replicate in form: schema1.table1,schema2.table2,...</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

---

**--preferred-path**

<table>
<thead>
<tr>
<th>Option</th>
<th>--preferred-path [393]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Config File Options</strong></td>
<td>preferred-path [393]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Additional command path</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>filename</td>
</tr>
</tbody>
</table>

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 Clustering, including MySQL, Ruby, Java, and backup/restore tools such as Percona Xtrabackup.

One or more paths can be specified by separating each directory with a colon. For example:

```shell
tpm ...
```

The `--preferred-path [393]` 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**

<table>
<thead>
<tr>
<th>Option</th>
<th>--prefetch-enabled [393]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Config File Options</strong></td>
<td>prefetch-enabled [393]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Should the replicator service be setup as a prefetch applier</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

---

**--prefetch-max-time-ahead**

<table>
<thead>
<tr>
<th>Option</th>
<th>--prefetch-max-time-ahead [393]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Config File Options</strong></td>
<td>prefetch-max-time-ahead [393]</td>
</tr>
</tbody>
</table>
### The tpm Deployment Command

<table>
<thead>
<tr>
<th>Description</th>
<th>Maximum number of seconds that the prefetch applier can get in front of the standard applier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value Type</td>
<td>numeric</td>
</tr>
</tbody>
</table>

---

**--prefetch-min-time-ahead**

<table>
<thead>
<tr>
<th>Option</th>
<th>--prefetch-min-time-ahead [394]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>prefetch-min-time-ahead [394]</td>
</tr>
<tr>
<td>Description</td>
<td>Minimum number of seconds that the prefetch applier must be in front of the standard applier</td>
</tr>
<tr>
<td>Value Type</td>
<td>numeric</td>
</tr>
</tbody>
</table>

---

**--prefetch-schema**

<table>
<thead>
<tr>
<th>Option</th>
<th>--prefetch-schema [394]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>prefetch-schema [394]</td>
</tr>
<tr>
<td>Description</td>
<td>Schema to watch for timing prefetch progress</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
<tr>
<td>Default</td>
<td>tungsten_</td>
</tr>
<tr>
<td>Valid Values</td>
<td>tungsten_</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**--prefetch-sleep-time**

<table>
<thead>
<tr>
<th>Option</th>
<th>--prefetch-sleep-time [394]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>prefetch-sleep-time [394]</td>
</tr>
<tr>
<td>Description</td>
<td>How long to wait when the prefetch applier gets too far ahead</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

---

**--privileged-master**

<table>
<thead>
<tr>
<th>Option</th>
<th>--privileged-master [394]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>privileged-master [394]</td>
</tr>
<tr>
<td>Description</td>
<td>Does the login for the master database service have superuser privileges</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

---

**--privileged-slave**

<table>
<thead>
<tr>
<th>Option</th>
<th>--privileged-slave [394]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>privileged-slave [394]</td>
</tr>
<tr>
<td>Description</td>
<td>Does the login for the slave database service have superuser privileges</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

---

**--profile-script**

<table>
<thead>
<tr>
<th>Option</th>
<th>--profile-script [394]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>profile-script [394]</td>
</tr>
<tr>
<td>Description</td>
<td>Append commands to include env.sh in this profile script</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

---

**--protect-configuration-files**

<table>
<thead>
<tr>
<th>Option</th>
<th>--protect-configuration-files [394]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>protect-configuration-files [394]</td>
</tr>
<tr>
<td>Description</td>
<td>When enabled, configuration files are protected to be only readable and updatable by the configured user</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>
### The `tpm` Deployment Command

<table>
<thead>
<tr>
<th>Valid Values</th>
<th>Make configuration files readable by any user</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>

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
cat /opt/continuent/tungsten/tungsten-replicator/conf/
```

When disabled, the files are readable by all users:

```shell
ls -al /opt/continuent/tungsten/tungsten-replicator/conf/
```

### 9.8.15. `tpm` Options

#### `--redshift-dbname`

<table>
<thead>
<tr>
<th>Option</th>
<th><code>--redshift-dbname [395]</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Aliases</strong></td>
<td><code>--repl-redshift-dbname [395]</code></td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td><code>redshift-dbname [395], repl-redshift-dbname [395]</code></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Name of the Redshift database to replicate into</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

#### `--relay-directory`

<table>
<thead>
<tr>
<th>Option</th>
<th><code>--relay-directory [395]</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Aliases</strong></td>
<td><code>--repl-relay-directory [395]</code></td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td><code>relay-directory [395], repl-relay-directory [395]</code></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Directory for logs transferred from the master</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
<tr>
<td><strong>Default</strong></td>
<td><code>{home directory}/relay</code></td>
</tr>
<tr>
<td><strong>Valid Values</strong></td>
<td><code>{home directory}/relay</code></td>
</tr>
<tr>
<td></td>
<td><code>{home directory}/relay</code></td>
</tr>
</tbody>
</table>

#### `--relay-enabled`

<table>
<thead>
<tr>
<th>Option</th>
<th><code>--relay-enabled [395]</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Config File Options</strong></td>
<td><code>relay-enabled [395]</code></td>
</tr>
<tr>
<td>Description</td>
<td>Should the replicator service be setup as a relay master</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

--relay-source

<table>
<thead>
<tr>
<th>Option</th>
<th>--relay-source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>dataservice-relay-source, master-dataservice</td>
</tr>
<tr>
<td>Config File Options</td>
<td>dataservice-relay-source, master-dataservice, relay-source</td>
</tr>
<tr>
<td>Description</td>
<td>Dataservice name to use as a relay source</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

--replace-tls-certificate

<table>
<thead>
<tr>
<th>Option</th>
<th>--replace-tls-certificate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>replace-tls-certificate</td>
</tr>
<tr>
<td>Description</td>
<td>Replace the TLS certificate</td>
</tr>
</tbody>
</table>

Replace the TLS certificate

--replication-host

<table>
<thead>
<tr>
<th>Option</th>
<th>--replication-host</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>datasource-host, repl-datasource-host</td>
</tr>
<tr>
<td>Config File Options</td>
<td>datasource-host, repl-datasource-host, replication-host</td>
</tr>
<tr>
<td>Description</td>
<td>Hostname of the datasource</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

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 slave (relay) connection.

--replication-password

<table>
<thead>
<tr>
<th>Option</th>
<th>--replication-password</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>datasource-password, repl-datasource-password</td>
</tr>
<tr>
<td>Config File Options</td>
<td>datasource-password, repl-datasource-password, replication-password</td>
</tr>
<tr>
<td>Description</td>
<td>Database password</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

The password to be used when connecting to the database using the corresponding --replication-user.

--replication-port

<table>
<thead>
<tr>
<th>Option</th>
<th>--replication-port</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>datasource-port, repl-datasource-port</td>
</tr>
<tr>
<td>Config File Options</td>
<td>datasource-port, repl-datasource-port, replication-port</td>
</tr>
<tr>
<td>Description</td>
<td>Database network port</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

Valid Values

- 1521: Oracle Default
- 27017: Elasticsearch Default
- 27017: Kafka Default
- 27017: MongoDB Default
- 3306: MySQL Default
- 5432: PostgreSQL Default
- 5433: Vertica Default
The network port used to connect to the database server. The default port used depends on the database being configured.

--replication-user

<table>
<thead>
<tr>
<th>Option</th>
<th>--replication-user [397]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--datasource-user [397], --repl-datasource-user [397]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>datasource-user [397], repl-datasource-user [397], replication-user [397]</td>
</tr>
<tr>
<td>Description</td>
<td>User for database connection</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

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

--reset

<table>
<thead>
<tr>
<th>Option</th>
<th>--reset [397]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>reset [397]</td>
</tr>
<tr>
<td>Description</td>
<td>Clear the current configuration before processing any arguments</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

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

--rmi-port

<table>
<thead>
<tr>
<th>Option</th>
<th>--rmi-port [397]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-rmi-port [397]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>repl-rmi-port [397], rmi-port [397]</td>
</tr>
<tr>
<td>Description</td>
<td>Replication RMI listen port</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

--rmi-user

<table>
<thead>
<tr>
<th>Option</th>
<th>--rmi-user [397]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>rmi-user [397]</td>
</tr>
<tr>
<td>Description</td>
<td>The username for RMI authentication</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

--role

<table>
<thead>
<tr>
<th>Option</th>
<th>--role [397]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-role [397]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>repl-role [397], role [397]</td>
</tr>
<tr>
<td>Description</td>
<td>What is the replication role for this service?</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
<tr>
<td>Valid Values</td>
<td>master</td>
</tr>
<tr>
<td></td>
<td>relay</td>
</tr>
<tr>
<td></td>
<td>slave</td>
</tr>
</tbody>
</table>

--router-gateway-port

<table>
<thead>
<tr>
<th>Option</th>
<th>--router-gateway-port [397]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>router-gateway-port [397]</td>
</tr>
</tbody>
</table>
### 9.8.16. Tpm Options

#### --router-jmx-port

<table>
<thead>
<tr>
<th>Description</th>
<th>The router gateway port</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### Option: --router-jmx-port

<table>
<thead>
<tr>
<th>Config File Options</th>
<th>router-jmx-port [398]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The router jmx port</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### Configuration File Options: router-jmx-port [398]

#### --security-directory

<table>
<thead>
<tr>
<th>Description</th>
<th>Storage directory for the Java security/encryption files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### Option: --security-directory

<table>
<thead>
<tr>
<th>Config File Options</th>
<th>security-directory [398]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Storage directory for the Java security/encryption files</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### --service-alias

<table>
<thead>
<tr>
<th>Description</th>
<th>Replication alias of this dataservice</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### Option: --service-alias

<table>
<thead>
<tr>
<th>Aliases</th>
<th>--dataservice-service-alias [398]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>dataservice-service-alias [398], service-alias [398]</td>
</tr>
<tr>
<td>Description</td>
<td>Replication alias of this dataservice</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### Configuration File Options: dataservice-service-alias [398], service-alias [398]

#### --service-name

Set the service name

#### Option: --service-name [398]

<table>
<thead>
<tr>
<th>Description</th>
<th>Set the service name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### --service-type

<table>
<thead>
<tr>
<th>Description</th>
<th>What is the replication service type?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### Option: --service-type [398]

<table>
<thead>
<tr>
<th>Aliases</th>
<th>--repl-service-type [398]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>repl-service-type [398], service-type [398]</td>
</tr>
<tr>
<td>Description</td>
<td>What is the replication service type?</td>
</tr>
<tr>
<td>Valid Values</td>
<td>local, remote</td>
</tr>
</tbody>
</table>

#### --skip-statemap

<table>
<thead>
<tr>
<th>Description</th>
<th>Do not copy the cluster-home/conf/statemap.properties from the previous install</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### Option: --skip-statemap [398]

<table>
<thead>
<tr>
<th>Config File Options</th>
<th>skip-statemap [398]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Do not copy the cluster-home/conf/statemap.properties from the previous install</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### --slaves

<table>
<thead>
<tr>
<th>Option</th>
<th>--slaves [398]</th>
</tr>
</thead>
</table>

### The tpm Deployment Command

<table>
<thead>
<tr>
<th>Aliases</th>
<th>--dataservice-slaves [398]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>dataservice-slaves [398], slaves [398]</td>
</tr>
<tr>
<td>Description</td>
<td>What are the slaves for this dataservice?</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Option</th>
<th>--start [399]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>start [399]</td>
</tr>
<tr>
<td>Description</td>
<td>Start the services after configuration</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Option</th>
<th>--start-and-report [399]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>start-and-report [399]</td>
</tr>
<tr>
<td>Description</td>
<td>Start the services and report out the status after configuration</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Option</th>
<th>--svc-allow-any-remote-service [399]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-svc-allow-any-remote-service [399]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>repl-svc-allow-any-remote-service [399], svc-allow-any-remote-service [399]</td>
</tr>
<tr>
<td>Description</td>
<td>Replicate from any service</td>
</tr>
<tr>
<td>Value Type</td>
<td>boolean</td>
</tr>
<tr>
<td>Valid Values</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td>true</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Option</th>
<th>--svc-applier-block-commit-interval [399]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-svc-applier-block-commit-interval [399]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>repl-svc-applier-block-commit-interval [399], svc-applier-block-commit-interval [399]</td>
</tr>
<tr>
<td>Description</td>
<td>Minimum interval between commits</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
<tr>
<td>Valid Values</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>#d Number of days</td>
</tr>
<tr>
<td></td>
<td>#h Number of hours</td>
</tr>
<tr>
<td></td>
<td>#m Number of minutes</td>
</tr>
<tr>
<td></td>
<td>#s Number of seconds</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Option</th>
<th>--svc-applier-block-commit-size [399]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-svc-applier-block-commit-size [399]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>repl-svc-applier-block-commit-size [399], svc-applier-block-commit-size [399]</td>
</tr>
<tr>
<td>Description</td>
<td>Applier block commit size (min 1)</td>
</tr>
<tr>
<td>Value Type</td>
<td>numeric</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Option</th>
<th>--svc-applier-filters [399]</th>
</tr>
</thead>
</table>
### The `tpm` Deployment Command

<table>
<thead>
<tr>
<th>Alias/Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>repl-svc-applier-filters</td>
<td>Replication service applier filters</td>
<td>string</td>
</tr>
<tr>
<td>repl-svc-extractor-filters</td>
<td>Replication service extractor filters</td>
<td>string</td>
</tr>
<tr>
<td>repl-svc-fail-on-zero-row-update</td>
<td>How should the replicator behave when a Row-Based Replication UPDATE does not affect any rows.</td>
<td>string</td>
</tr>
<tr>
<td>repl-svc-parallelization-type</td>
<td>Method for implementing parallel apply</td>
<td>string</td>
</tr>
<tr>
<td>repl-svc-remote-filters</td>
<td>Replication service remote download filters</td>
<td>string</td>
</tr>
<tr>
<td>repl-svc-reposition-on-source-id-change</td>
<td>The master will come ONLINE from the current position if the stored source_id does not match the value in the static properties</td>
<td>string</td>
</tr>
<tr>
<td>repl-svc-shard-default-db</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Config File Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>repl-svc-shard-default-db</code></td>
<td>Mode for setting the shard ID from the default db</td>
</tr>
<tr>
<td><code>svc-shard-default-db</code></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value Type</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>relaxed</td>
</tr>
<tr>
<td></td>
<td>stringent</td>
</tr>
</tbody>
</table>

#### `--svc-table-engine`

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--svc-table-engine</code></td>
<td>Replication service table engine</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value Type</th>
<th>Default</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>innodb</td>
<td>innodb</td>
</tr>
</tbody>
</table>

#### `--svc-thl-filters`

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--svc-thl-filters</code></td>
<td>Replication service THL filters</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value Type</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>innodb</td>
</tr>
</tbody>
</table>

---

### 9.8.17. T tpm Options

#### `--target-dataservice`

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--target-dataservice</code></td>
<td>Dataservice to use to determine the value of host configuration</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value Type</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

#### `--temp-directory`

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--temp-directory</code></td>
<td>Temporary Directory</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value Type</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

#### `--template-file-help`

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--template-file-help</code></td>
<td>Display the keys that may be used in configuration template files</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value Type</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

#### `--template-search-path`

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--template-search-path</code></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value Type</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>
## The tpm Deployment Command

<table>
<thead>
<tr>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adds a new template search path for configuration file generation</td>
<td>filename</td>
</tr>
</tbody>
</table>

### --thl-directory

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>--thl-directory [402]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replicator log directory</td>
<td>string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Default</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>{home directory}/thl</td>
<td>{home directory}/thl</td>
</tr>
</tbody>
</table>

### --thl-do-checksum

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>--thl-do-checksum [402]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute checksum operations on THL log files</td>
<td>string</td>
</tr>
</tbody>
</table>

### --thl-interface

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>--thl-interface [402]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Listen interface to use for THL operations</td>
<td>string</td>
</tr>
</tbody>
</table>

### --thl-log-connection-timeout

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>--thl-log-connection-timeout [402]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of seconds to wait for a connection to the THL log</td>
<td>numeric</td>
</tr>
</tbody>
</table>

### --thl-log-file-size

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>--thl-log-file-size [402]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>File size in bytes for THL disk logs</td>
<td>numeric</td>
</tr>
</tbody>
</table>

### --thl-log-fsync

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>--thl-log-fsync [402]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fsync THL records on commit. More reliable operation but adds latency to replication when using low-performance storage</td>
<td></td>
</tr>
</tbody>
</table>

---

402
### The tpm Deployment Command

<table>
<thead>
<tr>
<th>Value Type</th>
<th>string</th>
</tr>
</thead>
</table>

#### --thl-log-retention

<table>
<thead>
<tr>
<th>Option</th>
<th>--thl-log-retention [403]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-thl-log-retention [403]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>repl-thl-log-retention [403], thl-log-retention [403]</td>
</tr>
<tr>
<td>Description</td>
<td>How long do you want to keep THL files.</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
<tr>
<td>Valid Values</td>
<td></td>
</tr>
<tr>
<td>#d</td>
<td>Number of days</td>
</tr>
<tr>
<td>#h</td>
<td>Number of hours</td>
</tr>
<tr>
<td>#m</td>
<td>Number of minutes</td>
</tr>
<tr>
<td>#s</td>
<td>Number of seconds</td>
</tr>
<tr>
<td>7d</td>
<td>7 days</td>
</tr>
</tbody>
</table>

#### --thl-port

<table>
<thead>
<tr>
<th>Option</th>
<th>--thl-port [403]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-thl-port [403]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>repl-thl-port [403], thl-port [403]</td>
</tr>
<tr>
<td>Description</td>
<td>Port to use for THL Operations</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### --thl-protocol

<table>
<thead>
<tr>
<th>Option</th>
<th>--thl-protocol [403]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-thl-protocol [403]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>repl-thl-protocol [403], thl-protocol [403]</td>
</tr>
<tr>
<td>Description</td>
<td>Protocol to use for THL communication with this service</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### --topology

<table>
<thead>
<tr>
<th>Option</th>
<th>--topology [403]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--dataservice-topology [403]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>dataservice-topology [403], topology [403]</td>
</tr>
<tr>
<td>Description</td>
<td>Replication topology for the dataservice Valid values are star,cluster-slave,master-slave,fan-in,clustered,cluster-alias,all-masters,direct</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### --track-schema-changes

<table>
<thead>
<tr>
<th>Option</th>
<th>--track-schema-changes [403]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>track-schema-changes [403]</td>
</tr>
<tr>
<td>Description</td>
<td>This will enable filters that track DDL statements and write the resulting change to files on slave hosts. The feature is intended for use in some batch deployments.</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

### 9.8.18. U tpm Options

#### --user

<table>
<thead>
<tr>
<th>Option</th>
<th>--user [403]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>user [403]</td>
</tr>
<tr>
<td>Description</td>
<td>System User</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

### 9.8.19. V tpm Options

--vertica-dname

<table>
<thead>
<tr>
<th>Option</th>
<th>--vertica-dname [404]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-vertica-dname [404]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>repl-vertica-dname [404], vertica-dname [404]</td>
</tr>
<tr>
<td>Description</td>
<td>Name of the database to replicate into</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

### 9.8.20. W tpm Options

--witnesses

<table>
<thead>
<tr>
<th>Option</th>
<th>--witnesses [404]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--dataservice-witnesses [404]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>dataservice-witnesses [404], witnesses [404]</td>
</tr>
<tr>
<td>Description</td>
<td>Witness hosts for the dataservice</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>
Chapter 10. 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 stores, 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 master before writing the information into the THL, then no slave could ever access the table data, because the information would never be stored into the THL to be transferred to the slaves. However, if the filter was applied on the slave, then some slaves could replicate the table and database information, while other slaves could choose to ignore them. The filtering process also has an impact on other elements of the system. For example, filtering on the master 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 master, as shown in Figure 10.1, "Filters: Pipeline Stages on Masters".

Figure 10.1. Filters: Pipeline Stages on Masters

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 slave, the stages configured by default are shown in Figure 10.2, "Filters: Pipeline Stages on Slaves".
Replication Filters

Figure 10.2. Filters: Pipeline Stages on Slaves

- **remote-to-thl Stage**
  Remote THL information is read from a master 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 *colnames*.

For certain filter combinations this order can be significant. Some filters rely on the information provided by earlier filters.

### 10.1. 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` [400]
  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` [401]
  Apply the filter between the internal queue and when the transactions are written to the THL on the master (`q-to-thl`).

- `--repl-svc-remote-filters` [400]
  Apply the filter between reading from the remote THL server and writing to the local THL files on the slave (`remote-to-thl`).

- `--repl-svc-applier-filters` [399]
  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 slave, 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`: 
Replication Filters

A bad filter configuration will not stop the replicator from starting, but the replicator will be placed into the **OFFLINE** state.

To disable a previously enabled filter, 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
```

Multiple filters can be applied on any stage, and the filters will be processes 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.name      : bidiSlave  
name               : q-to-dbms  
processedMinSeqno  : 1  
taskCount          : 5  
Finished status command (stages)...  
```

The above output is from a standard slave replication installation showing the default filters enabled. The filter order can be determined by the number against each filter definition.

### 10.2. Enabling Additional Filters

The Tungsten Clustering 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:

```java
replicator.filter.rename=com.continuent.tungsten.replicator.filter.RenameFilter
replicator.filter.rename.definitionsFile=${replicator.home.dir}/samples/extensions/java/rename.csv
```
Repli... filtration. In th... 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 be 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`:

```plaintext
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`:

```plaintext
shell> ./tools/tspm configure \\
    --repl-svc-applier-filters=custom
```

### 10.3. 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:

```plaintext
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
taskId         : 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
taskId         : 1
NAME            VALUE
----            -----...
applier.class : com.continuent.tungsten.replicator.thl.THLApplier
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-dbs
processedMinSeqno: 1
taskId         : 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.

### 10.4. Filter Reference

The different filter types configured and available within the replicate 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.
Replication Filters

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 Clustering 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.

The list of filters and their basic description are provided in the table below.

<table>
<thead>
<tr>
<th>Filter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BidiRemoteSlaveFilter</td>
<td>Content</td>
<td>Suppresses events that originated on the local service [required for correct slave operation]</td>
</tr>
<tr>
<td>BuildAuditTable</td>
<td>Auditing</td>
<td>Builds an audit table of changes for specified schemas and tables</td>
</tr>
<tr>
<td>BuildIndexTable</td>
<td>Transformation</td>
<td>Merges multiple schemas into a single schema</td>
</tr>
<tr>
<td>CaseMappingFilter</td>
<td>Transformation</td>
<td>Transforms schema, table and column names to upper or lower case</td>
</tr>
<tr>
<td>CDCMetadataFilter</td>
<td>Auditing</td>
<td>Records change data capture for transactions to a separate change table [auditing]</td>
</tr>
<tr>
<td>ColumnNameFilter</td>
<td>Validation</td>
<td>Adds column name information to row-based replication events</td>
</tr>
<tr>
<td>ConsistencyCheckFilter</td>
<td>Validation</td>
<td>Adds consistency checking to events</td>
</tr>
<tr>
<td>DatabaseTransformFilter</td>
<td>Transformation</td>
<td>Transforms database or table names using regular expressions</td>
</tr>
<tr>
<td>DummyFilter</td>
<td>Miscellaneous</td>
<td>Allows for confirmation of filter configuration</td>
</tr>
<tr>
<td>EnumToStringFilter</td>
<td>Content</td>
<td>Updates enumerated values to their string-based equivalent</td>
</tr>
<tr>
<td>EventMetadataFilter</td>
<td>Content</td>
<td>Filters events based on metadata; used by default within sharding and multi-master topologies</td>
</tr>
<tr>
<td>HeartbeatFilter</td>
<td>Validation</td>
<td>Detects heartbeat events on masters or slaves</td>
</tr>
<tr>
<td>JavaScriptFilter</td>
<td>Miscellaneous</td>
<td>Enables filtering through custom JavaScripts</td>
</tr>
<tr>
<td>LoggingFilter</td>
<td>Logging</td>
<td>Logs filtered events through the standard replicator logging mechanism</td>
</tr>
<tr>
<td>MySQLSessionSupportFilter</td>
<td>Content</td>
<td>Filters transactions for session specific temporary tables and variables</td>
</tr>
</tbody>
</table>
### Replication Filters

<table>
<thead>
<tr>
<th>Filter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OptimizeUpdatesFilter</td>
<td>Optimization</td>
<td>Optimizes update statements where the current and updated value are the same</td>
</tr>
<tr>
<td>PrimaryKeyFilter</td>
<td>Optimization</td>
<td>Used during row-based replication to optimize updates using primary keys</td>
</tr>
<tr>
<td>PrintEventFilter</td>
<td>Logging</td>
<td>Outputs transaction event information to the replication logging system</td>
</tr>
<tr>
<td>RenameFilter</td>
<td>Transformation</td>
<td>Advanced schema, table and column-based renaming</td>
</tr>
<tr>
<td>ReplicateColumnsFilter</td>
<td>Content</td>
<td>Removes selected columns from row-based transaction data</td>
</tr>
<tr>
<td>ReplicateFilter</td>
<td>Content</td>
<td>Selects or ignores specification schemas and/or databases</td>
</tr>
<tr>
<td>SetToStringFilter</td>
<td>Content</td>
<td>Converts integer values in <code>SET</code> datatypes to string values</td>
</tr>
<tr>
<td>ShardFilter</td>
<td>Content</td>
<td>Used to enforce database schema sharding between specific masters</td>
</tr>
<tr>
<td>TimeDelayFilter</td>
<td>Miscellaneous</td>
<td>Delays transactions until a specific point in time has passed</td>
</tr>
</tbody>
</table>

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.

#### 10.4.1. `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.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>ansiquotes</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript Filter File</td>
<td>tungsten-replicator/support/filters-javascript/ansiquotes.js</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.ansiquotes</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>binlog-to-q</td>
</tr>
<tr>
<td><code>tpm</code> Option compatibility</td>
<td><code>--svc-extractor-filters [400]</code></td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Any event</td>
</tr>
</tbody>
</table>

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>

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:

```javascript
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:

```javascript
query = "SET sql_mode='';";
```
10.4.2. 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 multi-master and other bi-directional replication deployments.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>bidiSlave</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property prefix</td>
<td>replicator.filter.bidiSlave</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td></td>
</tr>
<tr>
<td>tpm Option compatibility</td>
<td></td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Any event</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>localServiceName</td>
<td>string</td>
<td>${local.service.name}</td>
<td>Local service name of the service that reads the binary log</td>
</tr>
<tr>
<td>allowBidiUnsafe</td>
<td>boolean</td>
<td>false</td>
<td>If true, allows statements that may be unsafe for bi-directional replication</td>
</tr>
<tr>
<td>allowAnyRemoteService</td>
<td>boolean</td>
<td>false</td>
<td>If true, allows statements from any remote service, not just the current service</td>
</tr>
</tbody>
</table>

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 slaves. For complex deployments, particularly those with bi-directional replication (including multi-master), the allowBidiUnsafe parameter may need to be enabled to allow certain statements to be re-executed.

10.4.3. 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.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>ansiquotes</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript Filter File</td>
<td>tungsten-replicator/support/filters-javascript/breadcrumbs.js</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.breadcrumbs</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>binlog-to-q</td>
</tr>
<tr>
<td>tpm Option compatibility</td>
<td>--svc-extractor-filters [400]</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Any event</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>server_id</td>
<td>numeric</td>
<td>[none]</td>
<td>MySQL server ID of the current host</td>
</tr>
</tbody>
</table>

To use the filter:

1. A table is created and populated with one more rows on the master server. For example:

```sql
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);
```
Now set an event to update the table regularly. For example, within MySQL an event can be created for this purpose:

```sql
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 works 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:

```java
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:

```java
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:

```java
if (event.getLastFrag())
{
  breadcrumb_offset = breadcrumb_offset + 1;
}
```

10.4.4. BuildAuditTable Filter

The BuildAuditTable filter populates a table with all the changes to a database so that the information can be tracked for auditing purposes.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>Not defined</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.filter.BuildAuditTable</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.bidiSlave</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>replicator.filter.bidiSlave</td>
</tr>
<tr>
<td>Option compatibility</td>
<td>not defined</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Row events</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableName</td>
<td>string</td>
<td></td>
<td>Name of the table where audit information will be stored</td>
</tr>
</tbody>
</table>
10.4.5. BuildIndexTable Filter

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>buildindextable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td><code>com.continuent.tungsten.replicator.filter.BuildIndexTable</code></td>
</tr>
<tr>
<td>Property prefix</td>
<td><code>replicator.filter.buildindextable</code></td>
</tr>
<tr>
<td>Stage compatibility</td>
<td></td>
</tr>
<tr>
<td>ipm Option compatibility</td>
<td></td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Row events</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>target_schema_name</td>
<td>string</td>
<td>test</td>
<td>Name of the schema where the new index information will be created</td>
</tr>
</tbody>
</table>

10.4.6. CaseMapping [CaseTransform] Filter

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>casetranform</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td><code>com.continuent.tungsten.replicator.filter.CaseMappingFilter</code></td>
</tr>
<tr>
<td>Property prefix</td>
<td><code>replicator.filter.casetransform</code></td>
</tr>
<tr>
<td>Stage compatibility</td>
<td></td>
</tr>
<tr>
<td>ipm Option compatibility</td>
<td></td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Any Event</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>to_upper_case</td>
<td>boolean</td>
<td>true</td>
<td>If true, converts object names to upper case; if false, converts them to lower case</td>
</tr>
</tbody>
</table>

10.4.7. CDCMetadata [CustomCDC] Filter

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>customcdc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td><code>com.continuent.tungsten.replicator.filter.CDCMetadataFilter</code></td>
</tr>
<tr>
<td>Property prefix</td>
<td><code>replicator.filter.customcdc</code></td>
</tr>
<tr>
<td>Stage compatibility</td>
<td></td>
</tr>
<tr>
<td>ipm Option compatibility</td>
<td></td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Row events</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cdcColumnsAtFront</td>
<td>boolean</td>
<td>false</td>
<td>If true, the additional CDC columns are added at the start of the table row. If false, they are added to the end of the table row</td>
</tr>
<tr>
<td>schemaNameSuffix</td>
<td>string</td>
<td></td>
<td>Specifies the schema name suffix. If defined, the tables are created in a schema matching schema name of the source transaction with the schema suffix appended</td>
</tr>
<tr>
<td>tableNameSuffix</td>
<td>string</td>
<td></td>
<td>Specifies the table name suffix for the CDC tables. If the schema suffix is not specified, this allows CDC tables to be created within the same schema</td>
</tr>
<tr>
<td>toSingleSchema</td>
<td>string</td>
<td></td>
<td>Creates and writes CDC data within a single schema</td>
</tr>
<tr>
<td>sequenceBeginning</td>
<td>numeric</td>
<td>1</td>
<td>Sets the sequence number of the CDC data. The sequence is used to identify individual changesets in the CDC</td>
</tr>
</tbody>
</table>
10.4.8. 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.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>colnames</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.filter.ColumnNameFilter</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.colnames</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>binlog-to-q</td>
</tr>
<tr>
<td>binlog Option compatibility</td>
<td>--svc-extractor-filters [400]</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Row events</td>
</tr>
<tr>
<td>Keeps Cached Data</td>
<td>Yes</td>
</tr>
<tr>
<td>Cached Refreshed When?</td>
<td>Emptied when going offline [212]; Updated when ALTER statement seen</td>
</tr>
<tr>
<td>Metadata Updated</td>
<td>Yes; tungsten_filter_columnname=true</td>
</tr>
</tbody>
</table>

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user</td>
<td>string</td>
<td>${replicator.global.extract.db.user}</td>
<td>The username for the connection to the database for looking up column definitions</td>
</tr>
<tr>
<td>password</td>
<td>string</td>
<td>${replicator.global.extract.db.password}</td>
<td>The password for the connection to the database for looking up column definitions</td>
</tr>
<tr>
<td>url</td>
<td>string</td>
<td>jdbc:mysql:thin://${replicator.global.extract.db.host}:${replicator.global.extract.db.port}/${replicator.schema}?createDatabase=false</td>
<td>JDBC URL of the database connection to use for looking up column definitions</td>
</tr>
<tr>
<td>addSignedFlag</td>
<td>boolean</td>
<td>true</td>
<td>Determines whether the signed flag information for columns should be added to the metadata for each column.</td>
</tr>
<tr>
<td>ignoreMissingTables</td>
<td>boolean</td>
<td>true</td>
<td>When true, tables that do not exist will not trigger metadata and column names to be added to the THL data.</td>
</tr>
</tbody>
</table>

**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 the 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:
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.

### 10.4.9. ConsistencyCheck Filter

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>Not defined</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.consistency.ConsistencyCheckFilter</td>
</tr>
<tr>
<td>Property prefix</td>
<td>Not defined</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>any</td>
</tr>
<tr>
<td>tpm Option compatibility</td>
<td>Any event</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Any event</td>
</tr>
<tr>
<td>Parameters</td>
<td>(none)</td>
</tr>
</tbody>
</table>

### 10.4.10. 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.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>convertstringfrommysql</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.ConvertStringFromMySQLFilter</td>
</tr>
<tr>
<td>Property prefix</td>
<td>Not defined</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>any</td>
</tr>
<tr>
<td>tpm Option compatibility</td>
<td>Row events only</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Row events only</td>
</tr>
<tr>
<td>Parameters</td>
<td>definitionsFile: string</td>
</tr>
</tbody>
</table>

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:

```json
{
  "__default": {
    "a": true,
  },
}
```
Warning

For column specific selection to work, the column names must be included within the THL. The \texttt{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 \texttt{true}, conversion will take place. A \texttt{false} disables conversion.

To enable conversion on a single column \texttt{DESCRIPTION} within the \texttt{SALES.INVOICE} schema/table while disabling conversion on all other columns:

\begin{verbatim}
{  "_default": {    "*": "false",  },  "SALES": {    "INVOICE": {      "DESCRIPTION": "true",    },  },}
\end{verbatim}

To convert all compatible columns in all tables within a schema:

\begin{verbatim}
{  "_default": {    "*": "false",  },  "SALES": {    "*": {      "*": "true",    },  },}
\end{verbatim}

A primary use case for this filter is for cluster-slave replication from a cluster to a datawarehouse. For more details, please see \textit{Replicating from a Cluster to a Datawarehouse} in [Tungsten Clustering (for MySQL) 5.3 Manual].

Source Cluster Example

For cluster-slave replication to a datawarehouse, the source cluster nodes must use ROW-based MySQL binary logging, and also must have two extractor filters enabled, \texttt{colnames} and \texttt{pkey}.

For example, on every cluster node the lines below would be added to the \texttt{/etc/tungsten/tungsten.ini} file in the service stanza, then \texttt{tpm update} would be executed:

\begin{verbatim}
repl-svc-extractor-filters=colnames,pkey
property=replicator.filter.pkey.addColumnsToDeletes=true
property=replicator.filter.pkey.addPkeyToInserts=true
\end{verbatim}

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-Slave Example

On the replication slave node, copy the \texttt{convertstringfrommysql.json} filter configuration sample file into the \texttt{/opt/replicator/share} directory then edit it to suit:

\begin{verbatim}
cp /opt/replicator/tungsten/tungsten-replicator/support/filters-config/convertstringfrommysql.json /opt/replicator/share/
vi /opt/replicator/share/convertstringfrommysql.json
\end{verbatim}

Once the \texttt{convertstringfrommysql} JSON configuration file has been edited, update the \texttt{/etc/tungsten/tungsten.ini} file to add and configure the \texttt{convertstringfrommysql} filter.

For example, configue a service named \texttt{omega} on \texttt{host6} to read from the cluster nodes defined by cluster-alias \texttt{alpha}.

\begin{verbatim}
[alpha]
topology=cluster-alias
master=host1
members=host1,host2,host3
thl-port=2112
\end{verbatim}
Replication Filters

For more details about configuring the target cluster-slave node, please see Section 3.4, "Replicating Data Out of a Cluster".

### 10.4.11. DatabaseTransform (dbtransform) Filter

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>dbtransform</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.filter.DatabaseTransformFilter</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.dbtransform</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td></td>
</tr>
<tr>
<td>tpm Option compatibility</td>
<td></td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Any event</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>transformTables</td>
<td>boolean</td>
<td>false</td>
<td>If set to true, forces the rename transformations to operate on tables, not databases</td>
</tr>
<tr>
<td>from_regex1</td>
<td>string</td>
<td>foo</td>
<td>The search regular expression to use when renaming databases or tables (group 1); corresponds to to_regex1</td>
</tr>
<tr>
<td>to_regex1</td>
<td>string</td>
<td>bar</td>
<td>The replace regular expression to use when renaming databases or tables (group 1); corresponds to from_regex1</td>
</tr>
<tr>
<td>from_regex2</td>
<td>string</td>
<td></td>
<td>The search regular expression to use when renaming databases or tables (group 2); corresponds to to_regex1</td>
</tr>
<tr>
<td>to_regex2</td>
<td>string</td>
<td></td>
<td>The replace regular expression to use when renaming databases or tables (group 2); corresponds to from_regex1</td>
</tr>
<tr>
<td>from_regex3</td>
<td>string</td>
<td></td>
<td>The search regular expression to use when renaming databases or tables (group 3); corresponds to to_regex1</td>
</tr>
<tr>
<td>to_regex3</td>
<td>string</td>
<td></td>
<td>The replace regular expression to use when renaming databases or tables (group 3); corresponds to from_regex1</td>
</tr>
</tbody>
</table>

### 10.4.12. 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.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>dbrename</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript Filter File</td>
<td>tungsten-replicator/support/filters-javascript/dbrename.js</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.dbrname</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>binlog-to-q</td>
</tr>
<tr>
<td>tpm Option compatibility</td>
<td>--svc-extractor-filters [400]</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Any event</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbsource</td>
<td>string</td>
<td>[none]</td>
<td>Source table name [database/table to be renamed]</td>
</tr>
<tr>
<td>dbtarget</td>
<td>string</td>
<td>[none]</td>
<td>New database/table name</td>
</tr>
</tbody>
</table>

To configure the filter you would add the following to your properties:

```properties
replicator.filter.dbrname=com.continuent.tungsten.replicator.filter.JavaScriptFilter
replicator.filter.dbrname.script=${replicator.home.dir}/samples/extensions/javascript/dbrename.js
```
Replication Filters

```
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:

```javascript
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);
                }
            }
        }
    }
}
```

**10.4.13. 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.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>dbselector</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript Filter File</td>
<td>tungsten-replicator/support/filters-javascript/dbselector.js</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.dbselector</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>binlog-to-q, q-to-thl, q-to-dbms</td>
</tr>
<tr>
<td>npm Option compatibility</td>
<td>--svc-extractor-filters [400], --svc-applier-filters [399]</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Any event</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>db</code></td>
<td>string</td>
<td>[none]</td>
<td>Database to be selected</td>
</tr>
</tbody>
</table>

Within the filter, statement changes look for the schema in the `StatementData` object and remove it from the array:

```javascript
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`:

```java
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--;
        }
    }
}
```

### 10.4.14. `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.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>dbupper</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript Filter File</td>
<td>tungsten-replicator/support/filters-javascript/dbupper.js</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.dbupper</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>binlog-to-q</td>
</tr>
<tr>
<td><code>tpm</code> Option compatibility</td>
<td>--svc-extractor-filters [400], --svc-applier-filters [399]</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Any event</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>from</code></td>
<td>string</td>
<td>(none)</td>
<td>Database name to be converted to uppercase</td>
</tr>
</tbody>
</table>

For example, within statement data:

```java
from = d.getDefaultSchema();
if (from != null) {
    to   = from.toUpperCase();
    d.setDefaultSchema(to);
}
```

### 10.4.15. `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 slave.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>dropcolumn</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript Filter File</td>
<td>tungsten-replicator/support/filters-javascript/dropcolumn.js</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.dropcolumn</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>binlog-to-q, q-to-dms</td>
</tr>
<tr>
<td><code>tpm</code> Option compatibility</td>
<td>--svc-extractor-filters [400], --svc-applier-filters [399]</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Any event</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>definitionsFile</code></td>
<td>Filename</td>
<td>~/dropcolumn.json</td>
<td>Location of the definitions file for dropping columns</td>
</tr>
</tbody>
</table>

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:

```bash
shell> tpm update --svc-extractor-filters=colnames,dropcolumn, \
    --property=replicator.filter.dropcolumn.definitionsFile=/opt/continuent/share/dropcolumn.json
```
Replication Filters

A sample configuration file is provided in `/opt/continuent/share/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 slave and master 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 master and slave, 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`.

10.4.16. **dropcomments.js** Filter

The **dropcomments** filter removes comments from statements within the event data.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th><code>dropcomments</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript Filter File</td>
<td>tungsten-replicator/support/filters-javascript/dropcomments.js</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.dropcomments</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>binlog-to-q, q-to-dbms</td>
</tr>
<tr>
<td><code>rpm</code> Option compatibility</td>
<td><code>--svc-extractor-filters [400], --svc-applier-filters [399]</code></td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Any event</td>
</tr>
</tbody>
</table>
Replication Filters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rowChanges</td>
<td>string</td>
<td>(none)</td>
<td>Name of the metadata field to be dropped</td>
</tr>
</tbody>
</table>

Metadata information can be processed at the event top-level:

```javascript
metaData = event.getDBMSEvent().getMetadata();
for(n = 0; n < metaData.size(); n++) {
    option = metaData.get(n);
    if(option.getName().compareTo(optionName)==0) {
        metaData.remove(n);
        break;
    }
}
```

10.4.18. **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. This is achieved by checking for statements, and then removing them from the event:

```javascript
data = event.getData();
```
Replication Filters

```java
for (i = 0; i < data.size(); i++)
{
    d = data.get(i);
    if (d instanceof com.continuent.tungsten.replicator.dbms.StatementData)
    {
        data.remove(i);
        i--;
    }
}
```

10.4.19. Dummy Filter

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>dummy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.filter.DummyFilter</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.dummy</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td></td>
</tr>
<tr>
<td><code>tpm</code> Option compatibility</td>
<td></td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Any event</td>
</tr>
<tr>
<td>Parameters</td>
<td>(none)</td>
</tr>
</tbody>
</table>

10.4.20. EnumToString Filter

The `EnumToString` filter translates `ENUM` datatypes within MySQL tables into their string equivalent within the THL.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>enumtostring</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.filter.EnumToStringFilter</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.enumtostring</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>binlog-to-q</td>
</tr>
<tr>
<td><code>tpm</code> Option compatibility</td>
<td>--repl-svc-extractor-filters [400]</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Row events</td>
</tr>
<tr>
<td>Metadata Updated</td>
<td>Yes; tungsten_filter_enumtostring=true</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user</td>
<td>string</td>
<td><code>${replicator.global.extract.db.user}</code></td>
<td>The username for the connection to the database for looking up column definitions</td>
</tr>
<tr>
<td>password</td>
<td>string</td>
<td><code>${replicator.global.extract.db.password}</code></td>
<td>The password for the connection to the database for looking up column definitions</td>
</tr>
<tr>
<td>url</td>
<td>string</td>
<td>jdbc:mysql:thin://${replicator.global.extract.db.host}:${replicator.global.extract.db.port}/${replicator.schema}?createDB=true</td>
<td>JDBC URL of the database connection to use for looking up column definitions</td>
</tr>
</tbody>
</table>

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 an `ENUM` column, `country`:

```
mysql> describe salesadv;
```

```sql
+----------+--------------------------------------+------+-----+---------+----------------+
| 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(18,2)                        | 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.
For the `country` column, the corresponding value in the TGL is 1. With the `EnumToString` filter enabled, the value is expanded to the corresponding string value:

```
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
```

The information is critical when applying the data to a dataserver that is not aware of the table definition, such as when replicating to Oracle or MongoDB.

The examples here also show the Section 10.4.36, “SetToString Filter” and Section 10.4.8, “ColumnName Filter” filters.

### 10.4.21. EventMetadata Filter

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>eventmetadata</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.filter.EventMetadataFilter</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.eventmetadata</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>Row events</td>
</tr>
<tr>
<td>tpn Option compatibility</td>
<td>Yes; tungsten_filter_settoString=true</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Row events</td>
</tr>
<tr>
<td>Metadata Updated</td>
<td>True</td>
</tr>
<tr>
<td>Parameters</td>
<td>(none)</td>
</tr>
</tbody>
</table>

### 10.4.22. `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
```

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>foreignkeychecks</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript Filter File</td>
<td>tungsten-replicator/support/filters-javascript/foreignkeychecks.js</td>
</tr>
</tbody>
</table>
### 10.4.23. Heartbeat Filter

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>[none]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.filter.HeartbeatFilter</td>
</tr>
<tr>
<td>Property prefix</td>
<td>[none]</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td></td>
</tr>
<tr>
<td>tcpm Option compatibility</td>
<td></td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Any event</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
</tr>
<tr>
<td>parameter</td>
<td>Type</td>
</tr>
<tr>
<td>heartbeatInterval</td>
<td>numeric</td>
</tr>
</tbody>
</table>

### 10.4.24. insertonly.js Filter

The `insertonly` filter filters events to only include ROW-based events using `INSERT`.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>insertonly</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript Filter File</td>
<td>tungsten-replicator/support/filters-javascript/insertonly.js</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.insertonly</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>q-to-dbs</td>
</tr>
<tr>
<td>tcpm Option compatibility</td>
<td>--svc-applier-filters [399]</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Row events</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
</tr>
<tr>
<td>parameter</td>
<td>Type</td>
</tr>
</tbody>
</table>

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:

```java
upCaseQuery = d.getQuery().trim().toUpperCase();
```

Then comparing the string for the corresponding statement types:

```java
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:

```java
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.
This is achieved by examining each row and removing row changes that do not match the `INSERT` action type:

```java
if (oneRowChange.getAction() != "INSERT") {
    rowChanges.remove(j);
    j--;
}
```

### 10.4.25. Logging Filter

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>logger</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.filter.LoggingFilter</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.logger</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td></td>
</tr>
<tr>
<td><code>tpm</code> Option compatibility</td>
<td></td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Any event</td>
</tr>
<tr>
<td>Parameters</td>
<td>(none)</td>
</tr>
</tbody>
</table>

### 10.4.26. MySQLSessionSupport {mysqlsessions} Filter

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>mysqlsessions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.filter.MySQLSessionSupportFilter</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.mysqlsession</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td></td>
</tr>
<tr>
<td><code>tpm</code> Option compatibility</td>
<td></td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Any event</td>
</tr>
<tr>
<td>Parameters</td>
<td>(none)</td>
</tr>
</tbody>
</table>

### 10.4.27. NetworkClient Filter

The `NetworkClientFilter` processes data in selected columns

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>networkclient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.filter.NetworkClientFilter</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.networkclient</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>Any</td>
</tr>
<tr>
<td><code>tpm</code> Option compatibility</td>
<td>--svc-extractor-filters [400], --svc-thl-filters [401], --svc-applier-filters [399]</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Row events</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>definitionsFile</td>
<td>pathname</td>
<td>${replicator.home.dir}/samples/extensions/java/network-client.json</td>
<td>The name of a file containing the definitions for how columns should be processed by filters</td>
</tr>
<tr>
<td>serverPort</td>
<td>number</td>
<td>3112</td>
<td>The network port to use when communicating with the network client</td>
</tr>
<tr>
<td>timeout</td>
<td>number</td>
<td>10</td>
<td>Timeout in seconds before treating the network client as failed when waiting to send or receive content.</td>
</tr>
</tbody>
</table>

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.
- The network server must be running before the replicator is started. If the network server cannot be found, replication will go offline.

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.

### 10.4.27.1. 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:

```json
{
    "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:

```json
{
    "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:

```json
{
    "String_to_HEX_v1": [
        {
            "table": "hextable",
            "schema": "hexdb",
            "columns": ["hexcol"]
        }
    ],
    "HEX_to_String_v1": [
        {
            "table": "hexagon",
            "schema": "sourcetext",
            "columns": ["itemtext"]
        }
    ]
}
```
10.4.27.2. 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:

  ```json
  {
  "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:

  ```json
  {
  "payload" : 22,
  "type" : "acknowledged",
  "protocol" : "v0_9",
  "service" : "firstrep",
  "return" : 0
  }
  ```

  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" : 8,
    "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**
Replication Filters

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:

```json
{
  "transformation" : "String_to_HEX_v1",
  "fragments" : 1,
  "type" : "filtered",
  "fragment" : 1,
  "return" : 0,
  "schema" : "hextable",
  "service" : "firstrep",
  "protocol" : "v0.9",
  "payload" : 8,
  "column" : "hexcol",
  "row" : 0
}
```

10.4.27.3. 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.

```perl
#!/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 $!

  print "********
Server waiting for client connection on port 3112
******


  # Waiting for a new client connection
  my $client_socket = $socket->accept();

  # Get 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

  my $data = "";
  while( $data = $client_socket->getline())
  {
    # Read up to 1024 characters from the connected client
    chomp($data);
    print "\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";
    }
  }
```
Replication Filters

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();

10.4.28. nocreatedbifnotexists.js Filter

The nocreatedbifnotexists filter removes statements that start with:

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>nocreatedbifnotexists</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript Filter File</td>
<td>tungsten-replicator/support/filters-javascript/nocreatedbifnotexists.js</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.nocreatedbifnotexists</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>q-to-dbms</td>
</tr>
<tr>
<td>ipm Option compatibility</td>
<td>--svc-applier-filters [399]</td>
</tr>
</tbody>
</table>
Replication Filters

<table>
<thead>
<tr>
<th>Data compatibility</th>
<th>Any event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
</tr>
</tbody>
</table>

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()` function that defines the string to be examined:

```java
function prepare() {
    beginning = "CREATE DATABASE IF NOT EXISTS";
}
```

Row based changes can be ignored, but for statement based events, the SQL is examined and the statement removed if the SQL starts with the text in the `beginning` variable:

```java
sql = d.getQuery();
if(sql.startsWith(beginning)) {
    data.remove(i);
    i--;
}
```

10.4.29. 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.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>optimizeupdates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.filter.OptimizeUpdatesFilter</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.optimizeupdates</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>rep</td>
</tr>
<tr>
<td>ipm Option compatibility</td>
<td>Data compatibility</td>
</tr>
<tr>
<td>Parameters</td>
<td>(none)</td>
</tr>
</tbody>
</table>

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:

```sql
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.
Replication Filters

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).

### 10.4.30. PrimaryKey Filter

The PrimaryKey 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 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

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>pkey</th>
</tr>
</thead>
</table>

#### Classname

com.continuent.tungsten.replicator.filter.PrimaryKeyFilter

#### Property prefix

replicator.filter.pkey

#### Stage compatibility

binlog-to-q

#### tpm Option compatibility

```
--repl-svc-extractor-filters [400]
```

#### Data compatibility

Row events

#### Keeps Cached Data

Yes

#### Cached Refreshed When?

Emptied when going OFFLINE [212]; Updated when ALTER statement seen

#### Metadata Updated

Yes; tungsten_filter_primarykey=true

#### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user</td>
<td>string</td>
<td>${replicator.global.extract.db.user}</td>
<td>The username for the connection to the database for looking up column definitions</td>
</tr>
<tr>
<td>password</td>
<td>string</td>
<td>${replicator.global.extract.db.password}</td>
<td>The password for the connection to the database for looking up column definitions</td>
</tr>
<tr>
<td>url</td>
<td>string</td>
<td>jdbc:mysql:thin://${replicator.global.extract.db.host}:${replicator.global.extract.db.port}/${replicator.schema}?createDatabase=true</td>
<td>JDBC URL of the database connection to use for looking up column definitions</td>
</tr>
<tr>
<td>addPkeyToInsert</td>
<td>boolean</td>
<td>false</td>
<td>If set to true, primary keys are added to INSERT operations. This setting is required for batch loading</td>
</tr>
<tr>
<td>addColumnsToDeletes</td>
<td>boolean</td>
<td>false</td>
<td>If set to true, full column metadata is added to DELETE operations. This setting is required for batch loading</td>
</tr>
</tbody>
</table>

#### 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 is the default behavior for UPDATE and DELETE operations.

```
SEQ # = 142 / FRAG # = 6 (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]
```
When the **PrimaryKey** is enabled, the key information has been optimized to only contain the actual primary keys are added to the row-based THL record:

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.

### 10.4.31. PrintEvent Filter

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>printevent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.filter.PrintEventFilter</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.printevent</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>replicator.filter.printevent</td>
</tr>
<tr>
<td>ipm Option compatibility</td>
<td>replicator.filter.printevent</td>
</tr>
</tbody>
</table>
10.4.32. 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.

Schema Renaming in 2.2.1 and later. The `rename` filter also performs schema renames on statement data only in Tungsten Replicator 2.2.1.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>rename</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.filter.RenameFilter</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.rename</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td></td>
</tr>
<tr>
<td>tpm Option compatibility</td>
<td></td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Row events; Schema names of Statement events in 2.2.1 and later.</td>
</tr>
</tbody>
</table>

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>definitionsFile</td>
<td>string</td>
<td>{replicator.home.dir}/samples/extensions/java/rename.csv</td>
<td>Location of the CSV file that contains the rename definitions.</td>
</tr>
</tbody>
</table>

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
dir> ./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.
- `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 all entries should match.
- `-` 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.
Replication Filters

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.

### 10.4.32.1 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
ewyork
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`. 

435
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

### 10.4.33. ReplicateColumns Filter

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>replicatecolumns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td><code>com.continuent.tungsten.replicator.filter.ReplicateColumnsFilter</code></td>
</tr>
<tr>
<td>Property prefix</td>
<td><code>replicator.filter.replicatecolumns</code></td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>Any</td>
</tr>
<tr>
<td><code>tpm</code> Option compatibility</td>
<td>Any</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Row events</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
</tr>
<tr>
<td>ignore</td>
<td>string</td>
</tr>
<tr>
<td>do</td>
<td>string</td>
</tr>
</tbody>
</table>

### 10.4.34. Replicate Filter

The **replicate** filter enables explicit inclusion or exclusion of tables and schemas. Each specification supports wildcards and multiple entries.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>replicate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td><code>com.continuent.tungsten.replicator.filter.ReplicateFilter</code></td>
</tr>
<tr>
<td>Property prefix</td>
<td><code>replicator.filter.replicate</code></td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>Any</td>
</tr>
<tr>
<td><code>tpm</code> Option compatibility</td>
<td>Any</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Any event</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
</tr>
<tr>
<td>ignore</td>
<td>string</td>
</tr>
<tr>
<td>do</td>
<td>string</td>
</tr>
</tbody>
</table>
Replication Filters

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.

10.4.35. 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.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>rowadddbname</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.filter.JavaScriptFilter</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.rowadddbname</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>Any</td>
</tr>
<tr>
<td>tpmm Option compatibility</td>
<td>Data compatibility</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td><code>adddbhash</code></td>
</tr>
<tr>
<td><code>addkey</code></td>
</tr>
<tr>
<td><code>fieldname</code></td>
</tr>
<tr>
<td><code>fieldhashname</code></td>
</tr>
</tbody>
</table>
The `rowdbname` 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 5.9, “Deploying the Vertica Applier”] and Amazon Redshift [see Section 5.8, “Deploying the Amazon Redshift Applier”].

### 10.4.36. SetToString Filter

The `SetToString` converts the `SET` column type from the internal representation to a string-based representation in the THL. This is achieved by accessing the extractor database, obtaining the table definitions, and modifying the THL data before it is written into the THL file.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>SetToString</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.filter.SetToStringFilter</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.setToString</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>binlog-to-q</td>
</tr>
<tr>
<td>RPM Option compatibility</td>
<td>--repl-svc-extractor-filters[400]</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Row events</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user</td>
<td>string</td>
<td><code>${replicator.global.extract.db.user}</code></td>
<td>The username for the connection to the database for looking up column definitions</td>
</tr>
<tr>
<td>password</td>
<td>string</td>
<td><code>${replicator.global.extract.db.password}</code></td>
<td>The password for the connection to the database for looking up column definitions</td>
</tr>
<tr>
<td>url</td>
<td>string</td>
<td>jdbc:mysql:thin://${replicator.global.extract.db.host}:${replicator.global.extract.db.port}/${replicator.schema}?createDatabase=false</td>
<td>JDBC URL of the database connection to use for looking up column definitions</td>
</tr>
</tbody>
</table>

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 |         |                |
```

438
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.

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:

The examples here also show the Section 10.4.20, “EnumToString Filter” and Section 10.4.8, “ColumnName Filter” filters.

### 10.4.37. Shard Filter

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>shardfilter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.filter.ShardFilter</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.shardfilter</td>
</tr>
</tbody>
</table>

#### Stage compatibility

- **tpm**
- **Option compatibility**
- **Data compatibility**
  - Any event

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enabled</td>
<td>boolean</td>
<td>false</td>
<td>If set to true, enables the shard filter</td>
</tr>
<tr>
<td>unknownShardPolicy</td>
<td>string</td>
<td>error</td>
<td>Select the filter policy when the shard unknown; valid values are accept, drop, warn, and error</td>
</tr>
<tr>
<td>unwantedShardPolicy</td>
<td>string</td>
<td>error</td>
<td>Select the filter policy when the shard is unwanted; valid values are accept, drop, warn, and error</td>
</tr>
<tr>
<td>enforcedHome</td>
<td>boolean</td>
<td>false</td>
<td>If true, enforce the home for the shard</td>
</tr>
<tr>
<td>allowWhitelisted</td>
<td>boolean</td>
<td>false</td>
<td>If true, allow explicitly whitelisted shards</td>
</tr>
<tr>
<td>autoCreate</td>
<td>boolean</td>
<td>false</td>
<td>If true, allow shard rules to be created automatically</td>
</tr>
</tbody>
</table>
10.4.38. **shardbyseqno.js Filter**

Shards within the replicator enable data to be parallelized when they are applied on the slave.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>shardbyseqno</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript Filter File</td>
<td>tungsten-replicator/support/filters-javascript/shardbyseqno.js</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.shardbyseqno</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>q-to-dbs</td>
</tr>
<tr>
<td>TPM Option compatibility</td>
<td>--svc-applier-filters [399]</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Any event</td>
</tr>
</tbody>
</table>

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>shards</td>
<td>numeric</td>
<td>[none]</td>
<td>Number of shards to be used by the applier</td>
</tr>
</tbody>
</table>

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 slaves.

10.4.39. **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.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>shardbytable</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript Filter File</td>
<td>tungsten-replicator/support/filters-javascript/shardbytable.js</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.shardbytable</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>q-to-dbs</td>
</tr>
<tr>
<td>TPM Option compatibility</td>
<td>--svc-applier-filters [399]</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Any event</td>
</tr>
</tbody>
</table>

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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;
}
```

10.4.40. **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.
Replication Filters

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>skipeventbytype</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.filter.SkipEventByTypeFilter</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.skipEventByType</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>any</td>
</tr>
<tr>
<td>tpm Option compatibility</td>
<td>--repl-svc-extractor-filters, --repl-svc-applier-filters</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Row events</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>definitionsFile</td>
<td>string</td>
<td>support/filters-config/skipEventByType.json</td>
<td>JSON file containing the definition of which events and which tables to skip</td>
</tr>
</tbody>
</table>

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": "allow",
      "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"
    }
  }
}
```

The configuration also supports single wildcard operations, for example:

```
{
  "__default": {
    "INSERT": "allow",
    "DELETE": "allow",
    "UPDATE": "allow"
  }
}
```
Replication Filters

```
"SALES" : {
  "*" : {
    "INSERT" : "deny",
    "DELETE" : "allow",
    "UPDATE" : "allow"
  }
}
```

Disables INSERT events for all tables within the SALES schema.

### 10.4.41. TimeDelay (delay) Filter

The TimeDelayFilter delays writing events to the THL and should be used only on slaves in the remote-to-thl stage. This delays writing the transactions into the THL files, but allows the application of the slave data to the database to continue without further intervention.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.filter.TimeDelayFilter</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.delay</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>remote-to-thl</td>
</tr>
<tr>
<td>tpm Option compatibility</td>
<td>--repl-svc-thl-filters</td>
</tr>
</tbody>
</table>

| Data compatibility | Any event |

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>delay</td>
<td>numeric</td>
<td>300</td>
<td>Number of seconds to delay transaction processing row</td>
</tr>
</tbody>
</table>

The TimeDelay 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 slave, or where data may need to be audited or checked before transactions are committed.

**Note**

For effective operation, master and slaves should be synchronized using NTP or a similar protocol.

To enable the TimeDelayFilter, use tpm command to enable the filter operation and the required delay. For example, to enable the delay for 900 seconds:

```
shell> ./tools/tpm update alpha --hosts=host1,host2,host3 \
    --repl-svc-applier-filters=delay \ 
    --property=replicator.filter.delay.delay=900
```

Time delay of transaction events should be performed with care, since the delay will prevent a slave from being up to date compared to the master. In the event of a node failure, an up to date slave is required to ensure that data is safe.

### 10.4.42. 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.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>tosingledb</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript Filter File</td>
<td>tungsten-replicator/support/filters-javascript/tosingledb.js</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.ansiquotes</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>q-to-dbms</td>
</tr>
<tr>
<td>tpm Option compatibility</td>
<td>--svc-applier-filters</td>
</tr>
</tbody>
</table>

| Data compatibility | Any event |

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>db</td>
<td>string</td>
<td>[none]</td>
<td>Database name into which to replicate all tables</td>
</tr>
<tr>
<td></td>
<td>skip</td>
<td>string</td>
<td>[none]</td>
<td>Comma-separated list of databases to be ignored</td>
</tr>
</tbody>
</table>
Replication Filters

A database can be optionally ignored through the `skip` parameter within the configuration:

```properties
--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:

```java
if(oldDb!=null && oldDb.compareTo(skip)!=0)
{
    d.setDefaultSchema(db);
}
```

### 10.4.43. `truncatetext.js` Filter

The `truncatetext` filter truncates a MySQL BLOB field.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>truncatetext</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript Filter File</td>
<td>tungsten-replicator/support/filters-javascript/truncatetext.js</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.truncatetext</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>binlog-to-q, q-to-dbms</td>
</tr>
<tr>
<td>tpm Option compatibility</td>
<td>--svc-extractor-filters [400], --svc-extractor-filters [400]</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Row events</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>length</code></td>
<td>numeric</td>
<td>[none]</td>
<td>Maximum size of truncated field [bytes]</td>
</tr>
</tbody>
</table>

The length is determined by the `length` parameter in the properties:

```java
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.

```java
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);
        }
    }
}
```

### 10.4.44. `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.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>zerodate2null</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript Filter File</td>
<td>tungsten-replicator/support/filters-javascript/zerodate2null.js</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.zerodate2null</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>q-to-dbms</td>
</tr>
<tr>
<td>tpm Option compatibility</td>
<td>--svc-applier-filters [399]</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Row events</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
</tr>
</thead>
</table>

443
Replication Filters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.

```java
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()
                }
            }
        }
    }
}
```

10.5. Standard JSON Filter Configuration

A number of the filters that are included as part of Tungsten Clustering 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:

```json
{
    "__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:

```json
{
    "__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 10.5.1, “Rule Handling and Processing” and Section 10.5.2, “Schema, Table, and Column Selection”.

### 10.5.1. 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 `__default` section instead.

The default rule is always processed and followed if there is no match for an explicit schema, table, or column definition.

### 10.5.2. Schema, Table, and Column Selection

The format of the JSON configuration 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:

```json
"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:

```json
"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:

```json
"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:

```json
```

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.

### 10.6. 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 custome filters with older releases of Tungsten Replicator or with the now deprecated, Open Source 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 10.6.2, “Installing Custom JavaScript Filters”](#).

### 10.6.1. 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:

```javascript
// 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)
```
Replication Filters

```java
// 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.

### 10.6.1.1. Implementable Functions

Each JavaScript filter must define one or more functions that are used to operate the filter process. The `filter()` 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()` 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()` 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()**
  
  The `release()` 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.

### 10.6.1.2. 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");
  ```
### Replication Filters

```java
    targetName = filterProperties.getString("dbtarget");
```

- **Generic Replicator Properties**

  General properties can be retrieved using the `properties` class and the `getString()` method:

```java
    master = properties.getString("replicator.thl.remote_uri");
```

#### 10.6.1.3. 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:

```java
    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:

```java
    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.

#### 10.6.1.4. Exposed Data Structures

Within the `filter()` 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 `filter()` 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` 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:

```
<table>
<thead>
<tr>
<th>ReplDBMSEvent</th>
<th>DBMSData</th>
<th>StatementData</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DBMSData</td>
<td>StatementData</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBMSData</td>
<td>RowChangeData</td>
<td>OneRowChange [451]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>StatementData</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DBMSData</td>
<td>RowChangeData</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>StatementData</td>
<td></td>
</tr>
</tbody>
</table>
```

448
A single event can contain both statement and row change information within the list of individual `DBMSData` events. An event or

10.6.1.4.1. 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.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getAppliedLatency()</td>
<td>Returns the latency of the embedded event. See Section E.2.6, “Terminology: Fields appliedLatency”</td>
</tr>
<tr>
<td>getData()</td>
<td>Returns an array of the <code>DBMSData</code> objects within the event</td>
</tr>
<tr>
<td>getDBMSEvent()</td>
<td>Returns the original <code>DBMSEvent</code> object</td>
</tr>
<tr>
<td>getEpochNumber()</td>
<td>Get the Epoch number of the stored event. See THL EPOCH# [504]</td>
</tr>
<tr>
<td>getEventId()</td>
<td>Returns the native event ID. See THL EVENTID [504]</td>
</tr>
<tr>
<td>getExtractedTstamp()</td>
<td>Returns the timestamp of the event.</td>
</tr>
<tr>
<td>getFragno()</td>
<td>Returns the fragment ID. See THL SEQNO [503]</td>
</tr>
<tr>
<td>get_lastFrag()</td>
<td>Returns true if the fragment is the last fragment in the event.</td>
</tr>
<tr>
<td>getSeqno()</td>
<td>Returns the native sequence number. See THL SEQNO [503]</td>
</tr>
<tr>
<td>getShardId()</td>
<td>Returns the shard ID for the event.</td>
</tr>
<tr>
<td>getSourceId()</td>
<td>Returns the source ID of the event. See THL SOURCEID [504]</td>
</tr>
<tr>
<td>setShardId()</td>
<td>Sets the shard ID for the event, which can be used by the filter to set the shard.</td>
</tr>
</tbody>
</table>

The primary method used is `getData()`, which returns an array of the individual `DBMSData` objects contain in the event:

```javascript
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:

```javascript
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:

```javascript
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.
Replication Filters

Updating the Shard ID

The `setSharedId()` 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 10.4.39, "shard-bytable.js Filter".

10.6.1.4.2. 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:

```java
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 10.6.1.4.3, "StatementData Objects". For row data, see Section 10.6.1.4.4, "RowChangeData Objects".

10.6.1.4.3. 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:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getQuery()</code></td>
<td>Returns the SQL statement</td>
</tr>
<tr>
<td><code>setQuery()</code></td>
<td>Updates the SQL statement</td>
</tr>
<tr>
<td><code>appendToQuery()</code></td>
<td>Appends a string to an existing query</td>
</tr>
<tr>
<td><code>getDefaultSchema()</code></td>
<td>Returns the default schema in which the statement was executed. The schema may be null for explicit or multi-schema queries.</td>
</tr>
<tr>
<td><code>setDefaultSchema()</code></td>
<td>Set the default schema for the SQL statement</td>
</tr>
<tr>
<td><code>getTimestamp()</code></td>
<td>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</td>
</tr>
</tbody>
</table>

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:

```java
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":

```java
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 10.4.12, "dbrename.js Filter".

### 10.6.1.4.4. 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 [451] objects provided.

The following methods are supported for the RowChangeData object:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>appendOneRowChange(rowChange)</td>
<td>Appends a single row change to the event, using the supplied OneRowChange [451] object.</td>
</tr>
<tr>
<td>getRowChanges()</td>
<td>Returns an array list of all the changes as OneRowChange [451] objects.</td>
</tr>
<tr>
<td>setRowChanges(rowChanges)</td>
<td>Sets the row changes within the event using the supplied list of OneRowChange objects.</td>
</tr>
</tbody>
</table>

For example, a typical row-based process will operate as follows:

```java
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 [451] 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:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getAction()</td>
<td>Returns the row action type, i.e. whether the row change is an INSERT, UPDATE or DELETE</td>
</tr>
<tr>
<td>getColumnSpec()</td>
<td>Returns the specification of each column within the row change</td>
</tr>
<tr>
<td>getColumnValues()</td>
<td>Returns the value of each column within the row change</td>
</tr>
<tr>
<td>getSchemaName()</td>
<td>Gets the schema name of the row change</td>
</tr>
<tr>
<td>getTableName()</td>
<td>Gets the table name of the row change</td>
</tr>
<tr>
<td>setColumnSpec()</td>
<td>Sets the column specification using an array of column specifications</td>
</tr>
<tr>
<td>setColumnValues()</td>
<td>Sets the column values</td>
</tr>
<tr>
<td>setSchemaName()</td>
<td>Sets the schema name</td>
</tr>
<tr>
<td>setTableName()</td>
<td>Sets the table name</td>
</tr>
</tbody>
</table>

### Changing Schema or Table Names

The schema, table and column names are exposed at different levels within the OneRowChange [451] 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:

```java
oneRowChange.setSchemaName('prefix_' + oneRowChange.getSchemaName());
```

To update a table name, the getTableName() and setTableName() can be used in the same manner:

```java
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:

```java
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:
Replication Filters

```java
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.

```java
columns = oneRowChange.getColumnSpec();
```

For each column specification a `ColumnSpec` object is returned, which supports the following methods:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getIndex()</code></td>
<td>Gets the index of the column within the row change</td>
</tr>
<tr>
<td><code>getLength()</code></td>
<td>Gets the length of the column</td>
</tr>
<tr>
<td><code>getName()</code></td>
<td>Returns the column name if available</td>
</tr>
<tr>
<td><code>getType()</code></td>
<td>Gets the type number of the column</td>
</tr>
<tr>
<td><code>getTypeDescription()</code></td>
<td></td>
</tr>
<tr>
<td><code>isBlob()</code></td>
<td>Returns true if the column is a blob</td>
</tr>
<tr>
<td><code>isNotNull()</code></td>
<td>Returns true if the column is configured as NOT NULL</td>
</tr>
<tr>
<td><code>isUnsigned()</code></td>
<td>Returns true if the column is unsigned.</td>
</tr>
<tr>
<td><code>setBlob()</code></td>
<td>Set the column blob specification</td>
</tr>
<tr>
<td><code>setIndex()</code></td>
<td>Set the column index order</td>
</tr>
<tr>
<td><code>setLength()</code></td>
<td>Returns the column length</td>
</tr>
<tr>
<td><code>setName()</code></td>
<td>Set the column name</td>
</tr>
<tr>
<td><code>setNotNull()</code></td>
<td>Set whether the column is configured as NOT NULL</td>
</tr>
<tr>
<td><code>setSigned()</code></td>
<td>Set whether the column data is signed</td>
</tr>
<tr>
<td><code>setType()</code></td>
<td>Set the column type</td>
</tr>
<tr>
<td><code>setTypeDescription()</code></td>
<td></td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Type</th>
<th>Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>CHAR or VARCHAR</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>TEXT or BLOB</td>
<td>2004</td>
<td>Use <code>isBlob()</code> to identify if the column is a blob or not</td>
</tr>
<tr>
<td>TIME</td>
<td>92</td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>91</td>
<td></td>
</tr>
<tr>
<td>DATETIME or TIMESTAMP</td>
<td>92</td>
<td></td>
</tr>
<tr>
<td>DOUBLE</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

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.
Replication Filters

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:

```java
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() vs. getColumnValues()

<table>
<thead>
<tr>
<th>getColumnSpec()</th>
<th>msgid</th>
<th>message</th>
<th>msgdate</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0]</td>
<td>1</td>
<td>Hello New York!</td>
<td>Thursday, June 13, 2013</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getValue()</td>
<td>Get the current column value</td>
</tr>
<tr>
<td>setValue()</td>
<td>Set the column value to the supplied value</td>
</tr>
<tr>
<td>setValueNull()</td>
<td>Set the column value to NULL</td>
</tr>
</tbody>
</table>

For example, within the `zeroDate2null` sample, dates with a zero value are set to NULL using the following code:

```java
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.

### 10.6.2. 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’

#### 10.6.2.1. Step 1: Copy JavaScript files

By default, the software package will be contained in `/opt/continuent/software/tungsten-replicator-5.4.1-41`. 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:
10.6.2.2. 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-5.4.1-43/tungsten-replicator/samples/conf/filters/default`

10.6.2.3. 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
```

10.6.2.4. 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` property for the extractor replicator. If you are applying your custom filters to your applier, then use `svc-applier-filters` 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` state and the reason for the error will be provided within the replicator `trepsvc.log` log.
Chapter 11. Performance and Tuning

To help improve the performance of Tungsten Clustering, a number of guides and tuning techniques are provided in this chapter. This may involve parameters entirely within Tungsten Clustering, or changes and modifications to the parameters within the OS.

Tuning related to the Tungsten Replicator functionality

- **Section 11.1, “Block Commit” — Increasing performance of replication solutions making use of block commit.

Tuning related to the network performance

- **Section 11.2, “Improving Network Performance” — Increasing performance of networking between components by tuning OS parameters.

11.1. Block Commit

Introduced in 2.2.0. The commit size and interval settings were introduced in 2.2.0.

The replicator commits changes read from the THL and commits these changes in slaves 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`)"[399])
- When the commit timer reaches the specified commit interval (set by `--svc-applier-block-commit-interval`[399])]

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.a-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` option to `tpm`, or through the `blockCommitRowCount`. The block commit interval can be controlled using the `--repl-svc-applier-block-commit-interval` 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.

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-size=20 | \
    --repl-svc-applier-block-commit-interval=100s
```

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-interval=500
```

```
shell> ./tools/tpm update alpha | \
    --repl-svc-applier-block-commit-size=455
```
--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.

### 11.1.1. 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                 0.003
stage                     q-to-dbms
state                     extract
taskId                    0

Finished status command (tasks)...
```

### 11.2. 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:

```plaintext
# 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

# Maxium 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 SYMACKs for passive TCP connection.
et.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
# sonx4n defmtes the number of request_sock structures
# allocated per each listen call. The
# queue is persistent through the life of the listen socket.
net.core.sonx4n = 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.optmen_max = 25165824

# Increase the maximum total buffer-space allocatable
# This is measured in units of pages (4096 bytes)
net.ipv4.tcp_men = 65536 131072 262144
net.ipv4.udp_men = 65536 131072 262144

### Set the max OS send buffer size (wmem) and receive buffer
# size (rmen) 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.rmen_default = 25165824

# Maximum Socket Receive Buffer
net.core.rmen_max = 25165824

# Increase the read-buffer space allocatable (minimum size,
# initial size, and maximum size in bytes)
net.ipv4.tcp_rmen = 20480 12582912 25165824
net.ipv4.udp_rmen_min = 16384

# Default Socket Send Buffer
net.core.wmen_default = 25165824

# Maximum Socket Send Buffer
net.core.wmen_max = 25165824

# Increase the write-buffer-space allocatable
net.ipv4.tcp_wmen = 20480 12582912 25165824
net.ipv4.udp_wmen_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_reuse = 1

11.3. 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 see from the attached diagram, a slave pipeline might have a stage to read transactions to the master 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 slave. 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 once, 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 replica-
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
shells> ./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
shells> ./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 5.4.1 GA [28 October 2019]

Version End of Life: Not Yet Set

Release 5.4.1 contains both significant improvements as well as some needed bug fixes.

Improvements, new features and functionality

- Core Replicator
  - If using the dropcolumn filter during extraction, in conjunction with the Batch Applier (e.g., Replicating to Redshift, Hadoop, Vertica) writes would fail with a CSV mismatch error due to gaps in the THL Index.

  However, for JDBC applied data, the gaps are required to ensure the correct column mapping.

  To handle the two different requirements, a new property has been added to the filter to control whether or not to leave the THL index untouched (the default) or to re-order the Index ID's.

  If applying to Batch targets, then the following property should be added to your configuration. The property is not required for JDBC targets.

  ```
  --property=replicator.filter.dropcolumn.fillGaps=true
  ```

  Issues: CT-1025

Bug Fixes

- Command-line Tools
  - Fixed an issue that would prevent reading remote binary logs when using SSL.
    Issues: CT-958
  - Restored previously-removed log file symbolic links under $CONTINUENT_ROOT/service_logs/
    Issues: CT-1026
  - Fixed a bug where tpm diag would generate an empty zip file if the hostnames contain hyphens (-) or periods (.)
    Issues: CT-1032
  - Improve ability to find needed binaries for commands: tungsten_find_position, tungsten_find_seqno and tungsten_get_rtt
    Issues: CT-1054

A.2. Tungsten Replicator 5.4.0 GA [31 July 2019]

Version End of Life: Not Yet Set

Release 5.4.0 contains both significant improvements as well as some needed bug fixes. One of the main features of this release is MySQL 8 support.

Improvements, new features and functionality

- Command-line Tools
  - Two new utility scripts have been added to the release to help with setting the Replicator position:
    - `tungsten_find_position`, which assists with locating information in the THL based on the provided MySQL binary log event position and outputs a dsctl set command as output.
    - `tungsten_find_seqno`, which assists with locating information in the THL based on the provided sequence number and outputs a dsctl set command as output.
    Issues: CT-934

- Core Replicator
The replicator has been updated to support the new character sets supported by MySQL 5.7 and MySQL 8.0, including the UTF-8-mb4 series.

**Issues**: CT-700, CT-970

A new, beta-quality command has been included called prov-sl.sh which is intended to eventually replace the current tungsten_provision_slave script.

Currently, prov-sl.sh supports provisioning slaves using mysqldump and xtrabackup tools, and is MySQL 8-compatible.

The prov-sl.sh command is written in Bash, has less dependencies compared to the current version and is meant to fix a number of issues with the current version.

Backups are streamed from source to target so that an intermediate write to disk is not performed, resulting in faster provisioning times.

Logs are written to $CONTINUENT_ROOT/service_logs/prov-sl.log (i.e. /opt/continuent/service_logs/prov-sl.log).

For example, provision a slave from [source db] using mysqldump (default):

```
shell> prov-sl.sh -s {source db}
```

As another example, use xtrabackup for the backup method, with 10 parallel threads (default is 4), and ssh is listening on port 2222:

```
shell> prov-sl.sh -s {source db} -n xtrabackup -t 10 -p 2222
```

**Warning**

At the moment, prov-sl.sh does not support MultiMaster topologies when used with Tungsten Clustering, however it will be included in a future release.

**Issues**: CT-614, CT-723, CT-809, CT-855, CT-963

Upgraded the Drizzle driver to support MySQL 8 authentication protocols (SHA256, caching_sha2).

**Issues**: CT-914, CT-931, CT-966

The Redshift Applier now allows AWS authentication using IAM Roles. Previously authentication was possible via Access and Secret Key pairs only.

**Issues**: CT-980

For more information, see Section 5.8.2.1, “Redshift Preparation for Amazon Redshift Deployments”.

**Bug Fixes**

**Command-line Tools**

The --hosts option was not working with the diag sub-command of the tpm command on nodes installed using the INI method.

The corrected behavior is as follows:

- With Staging-method deployments, the tpm diag command continues to behave as before:
  - 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.

- With INI-method deployments, the new behavior is as follows:
  - The tpm diag command alone will obtain diagnostics from the local host only.
  - The tpm diag --hosts host1,host2,hostN command will obtain diagnostics from the specified host[s] only.

**Warning**

Limitation: the host list MUST include the local hostname or the command will fail.

**Issues**: CT-345

When using tpm with the INI method, the command would search multiple locations for suitable INI files. This could lead to multiple definitions of the same service, which could in turn lead to duplication of the installation process and occasional failures. If multiple INI files are found, a warning is now produced to highlight the potential for failures.
Issues: CT-626
• The `trepctl` command now properly handles the `-all-services` option for the `reset` sub-command.

Issues: CT-762
• The command `tpm reverse --ini-format` now outputs correctly (without the double-dashes and the trailing backslash).

Issues: CT-827, CT-877
• The `tpm` command now properly handles network interface names containing colons and/or dots.

Issues: CT-864
• When executing `mysqldump`, all Tungsten tools no longer use the `--add-drop-database` flag as it will prevent MySQL 8+ from restoring the dump.

Issues: CT-935
• Fixed a bug where `tpm diag` would generate an empty zip file if the hostnames contain hyphens (-) or periods (.)

Issues: CT-915, CT-932
• Core Replicator
• Added support for missing charset GB18030 to correct `WARN extractor.mysql.MysqlBinlog Unknown charset` errors.

Issues: CT-984
• Loading data into Redshift would fail with the following error if a row of data contained a specific control character (0x00 [NULL])

```
Missing newline: unexpected character 0x30 found at location nnn
```

Issues: CT-997
• Now properly extracting the Geometry datatype.

Issues: CT-1002
• The `ddl_map.json` file used by the `apply_schema_changes` filter was missing a rule to handle `ALTER TABLE` statements when replicating between MySQL and Redshift

Issues: CT-1011
• The `extract_schema_change` filter wasn't escaping " (double-quotes) and the generated JSON would then cause the applier to error with

```
pendingExceptionMessage: SyntaxError: missing } after property list
```

Issues: CT-1011
Appendix B. Prerequisites

Before you install Tungsten Clustering, 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

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Variant</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td>RedHat/CentOS</td>
<td>Primary platform</td>
<td>RHEL 4, 5, and 6 as well as CentOS 5.x and 6.x versions are fully supported. CentOS 7 is supported in 5.3.0 and higher.</td>
</tr>
<tr>
<td>Linux</td>
<td>Ubuntu</td>
<td>Primary platform</td>
<td>Ubuntu 9.x-17.x versions are fully supported.</td>
</tr>
<tr>
<td>Linux</td>
<td>Debian/Suse/Other</td>
<td>Secondary Platform</td>
<td>Other Linux platforms are supported but are not regularly tested. We will fix any bugs reported by customers.</td>
</tr>
<tr>
<td>Linux</td>
<td>Docker</td>
<td>Unsupported</td>
<td>Unsupported - Use at your own risk. Docker containers are not well-suited for Tungsten deployments.</td>
</tr>
<tr>
<td>Solaris</td>
<td></td>
<td>Secondary Platform</td>
<td>Solaris 10 is fully supported. OpenSolaris is not supported at this time.</td>
</tr>
<tr>
<td>Mac OS X</td>
<td></td>
<td>Secondary platform</td>
<td>Mac OS/X is used for development at Continuent and could also be used as a Staging Host for Staging Deployments only, but it is not certified for us as a cluster or replicator node within a topology.</td>
</tr>
<tr>
<td>Windows</td>
<td></td>
<td>Limited Support</td>
<td>Tungsten 1.3 and above will support Windows platforms for connectivity (Tungsten Connector and SQL Router) but may require manual configuration. Tungsten clusters do not run on Windows.</td>
</tr>
<tr>
<td>BSD</td>
<td></td>
<td>Limited Support</td>
<td>Tungsten 1.3 and above will support BSD for connectivity (Tungsten Connector and SQL Router) but may require manual configuration. Tungsten clusters do not run on BSD.</td>
</tr>
</tbody>
</table>

B.1.2. Database Support

<table>
<thead>
<tr>
<th>Database</th>
<th>Version</th>
<th>Support Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL</td>
<td>5.0, 5.1, 5.5, 5.6, 5.7 [without Geometry support in 5.3.0 or 6.0.0]; full support was provided in 5.3.1 and 6.0.1</td>
<td>Primary platform</td>
<td>Statement and row based replication is supported. MyISAM and InnoDB table types are fully supported; InnoDB tables are recommended.</td>
</tr>
<tr>
<td>MySQL</td>
<td>5.7</td>
<td>Primary platform</td>
<td>Support is provided for compatibility with MySQL 5.7 in 5.0 and later, but new datatypes (JSON, virtual columns) in MySQL 5.7 are not yet supported.</td>
</tr>
<tr>
<td>Percona</td>
<td>5.5, 5.6, 5.7</td>
<td>Primary platform</td>
<td></td>
</tr>
<tr>
<td>MariaDB</td>
<td>5.5, 10.0, 10.1</td>
<td>Primary platform</td>
<td></td>
</tr>
<tr>
<td>Oracle [CDC]</td>
<td>10g Release 2 [10.2.0.5], 11g</td>
<td>Primary Platform</td>
<td>Synchronous CDC is supported on Standard Edition only; Synchronous and Asynchronous are supported on Enterprise Editions</td>
</tr>
<tr>
<td>Oracle [Redo Reader]</td>
<td>9.2, 10g, 11g, 12c</td>
<td>Primary Platform</td>
<td>Redo Reader reads data direct from the Oracle redo logs.</td>
</tr>
<tr>
<td>Drizzle</td>
<td></td>
<td>Secondary Platform</td>
<td>Experimental support for Drizzle is available. Drizzle replication is not tested.</td>
</tr>
</tbody>
</table>
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 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.

When replicating from Oracle, the size of the THL will depend on the quantity of Change Data Capture (CDC) information generated. This can be managed by altering the intervals used to check for and extract the information.

A dedicated partition for THL or Tungsten Clustering 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

Tungsten Replicator is known to work with with the following Java versions and JVMs:

- Oracle JVM/JDK 7
- Oracle JVM/JDK 8
- OpenJDK 7
- OpenJDK 8

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:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Guidance</th>
<th>Amazon Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance Type</td>
<td>Instance sizes and types are dependent on the workload, but larger instances are recommended for transactional databases.</td>
<td>m4.xlarge or better</td>
</tr>
<tr>
<td>Instance Boot Volume</td>
<td>Use block, not ephemeral storage.</td>
<td>EBS</td>
</tr>
<tr>
<td>Instance Deployment</td>
<td>Use standard Linux distributions and bases. For ease of deployment and configuration, use Puppet.</td>
<td>Amazon Linux AMIs</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>tpm Option</th>
<th>tpm Value</th>
<th>MySQL ny.cnf Option</th>
<th>MySQL Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>/ (root)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MySQL Data</td>
<td>datasource/mysql-data-directory [373]</td>
<td>/volumes/mysql/data</td>
<td>datadir</td>
<td>/volumes/mysql/data</td>
</tr>
<tr>
<td>MySQL Binary Logs</td>
<td>datasource-log-directory [373]</td>
<td>/volumes/mysql/binlogs</td>
<td>log-bin</td>
<td>/volumes/mysql/binlogs/mysql-bin</td>
</tr>
<tr>
<td>Transaction History Logs [THL]</td>
<td>thl-directory [402]</td>
<td>/volumes/mysql/thl</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Recommended Replication Formats

- **MIXED** is recommended for MySQL master/slave topologies (e.g., either single clusters or primary/data-recovery setups).
- **ROW** is strongly recommended for multi-master 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 master server binary logs to the slave 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 Clustering does 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.
Prerequisites

- Once Tungsten Clustering has 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 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. What to Expect in the Future

Continuent does NOT have Docker containerization on the product roadmap at this time. That being said, we do intend to provide containerization support at some point in the future. Customer demand will contribute to the timing of the effort.

B.1.7.5. 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 Clustering™ software, and to install, transfer, and initiate the Tungsten Clustering™ 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 Clustering™ 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 Clustering™. 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`:

   ```bash
   shell> sudo adduser tungsten
   
   You can add the user to the *mysql* group adding the command-line option:
   ```
   ```bash
   shell> sudo usermod -G mysql -a tungsten
   
   2. Login as the *tungsten* user:
   ```
   ```bash
   shell> su - tungsten
   
   3. Create an SSH key file, but do not configure a password:

   ```bash
   tungsten:$ 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:
   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.

   ```bash
   shell> mkdir -p /opt/continuent/software/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
**Prerequisites**

- 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`
### 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 most 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
  127.0.1.1     host1
  127.0.1.2     host2
  ```

- 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 `192.168.1.69` 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 Host2
   ```

469
Prerequisites

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:

<table>
<thead>
<tr>
<th>Component</th>
<th>Source</th>
<th>Destination</th>
<th>Port</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database Service</td>
<td>Database Host</td>
<td>Database Host</td>
<td>7</td>
<td>Checking availability</td>
</tr>
<tr>
<td>#</td>
<td>#</td>
<td>#</td>
<td>2112</td>
<td>THL replication</td>
</tr>
<tr>
<td>#</td>
<td>#</td>
<td>#</td>
<td>10000-10001</td>
<td>Replication connection listener port</td>
</tr>
</tbody>
</table>

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 Clustering™, 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 Clustering™ 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:
Prerequisites

- Home Directory

The home directory of the tungsten user must be writable by that user.

B.3.4. Configure Software

Tungsten Clustering™ relies on the following software. Each host must use the same version of each tool.

<table>
<thead>
<tr>
<th>Software</th>
<th>Versions Supported</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>rsync</td>
<td>-</td>
<td>Check using rsync --help</td>
</tr>
<tr>
<td>Ruby</td>
<td>1.8.7, 1.9.3, or 2.0.0 to 2.4.0 a</td>
<td>JRuby is not supported</td>
</tr>
<tr>
<td>Ruby OpenSSL Module</td>
<td>-</td>
<td>Checking using ruby -ropenssl -e 'p &quot;works&quot;'</td>
</tr>
<tr>
<td>Ruby Gems</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Ruby io-console module</td>
<td>-</td>
<td>Install using gem install io-console b</td>
</tr>
<tr>
<td>Ruby net-ssh module</td>
<td>-</td>
<td>Install using gem install net-ssh c</td>
</tr>
<tr>
<td>Ruby net-scp module</td>
<td>-</td>
<td>Install using gem install net-scp d</td>
</tr>
<tr>
<td>GNU tar</td>
<td>-</td>
<td>gtar is required for Solaris due to limitations in the native tar command</td>
</tr>
<tr>
<td>Java Runtime Environment</td>
<td>Java SE 7 [or compatible],</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Java SE 8 [or compatible] is supported in 5.0.1 and higher</td>
<td></td>
</tr>
</tbody>
</table>

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 "11.0.10"
OpenJDK Runtime Environment (build 11.0.10+0.60)
OpenJDK 64-Bit Server VM (build 11.0.10+0.60, mixed mode)
```

Tungsten Clustering is known to work with Java using the following JVMs:

- Oracle JVM/JDK 8
- OpenJDK 7

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 7.14, “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.
Prerequisites

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 Ubuntu you can do this by editing the /etc/sudoers file using visudo and adding the following lines:

```
Defaults:tungsten !authenticate
... # Allow tungsten to run any command
tungsten ALL=(ALL) ALL
```

For a secure environment where sudo access is not permitted for all operations, a minimum configuration can be used:

```
tungsten ALL=(ALL)
```

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 tungsten_provision_slave, 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/sar, /usr/bin/which, *
/etc/init.d/mysql, /usr/bin/test, *
/apps/tungsten/continuent/tungsten/tungsten-replicator/scripts/xtrabackup.sh, *
/apps/tungsten/continuent/tungsten/tools/tpm, /usr/bin/innobackupex-1.5.1, *
/bin/cat, /bin/find
```

Within Red Hat Linux add the following line:

```
tungsten ALL=(root) NOPASSWD: ALL
```

For a secure environment where sudo access is not permitted for all operations, a minimum configuration can be used:

```
tungsten ALL=(root) NOPASSWD: /usr/bin/which, /etc/init.d/mysql
```

When using xtrabackup, or additional tools in the Tungsten toolkit, such as tungsten_provision_slave, additional commands must be added to the permitted list:

```
tungsten ALL=(root) NOPASSWD: /sbin/service, /usr/bin/innobackupex, /bin/rm, /bin/mv, /bin/chown, /bin/chmod, /usr/bin/scp, /bin/sar, /usr/bin/which, *
/etc/init.d/mysql, /usr/bin/test, *
/apps/tungsten/continuent/tungsten/tungsten-replicator/scripts/xtrabackup.sh, *
/apps/tungsten/continuent/tungsten/tools/tpm, /usr/bin/innobackupex-1.5.1, *
/bin/cat, /bin/find
```

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

Important

Tungsten best practice is to DISABLE SELinux.

If you are having problems, disable SELinux entirely and test fully. Once the system is fully operational and stable, then re-enable SELinux and see if the problem comes back. If so, then detailed configuration will be required. Such configuration is outside the scope of the Continuent Support Agreement.

To determine the current 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
```

Warning

Disclaimer: Continuent support staff will do our best to help but can't guarantee results. SELinux context errors are extremely difficult to debug, and difficult to determine the appropriate command to give the correct context. A simple
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 Clustering 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 Clustering™. The replication service will be completely handled by Tungsten Clustering™, and the normal replication, management and monitoring techniques will not provide you with the information you need.

---

### B.4.1. MySQL Version Support

<table>
<thead>
<tr>
<th>Database</th>
<th>Version</th>
<th>Support Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL</td>
<td>5.0, 5.1, 5.5, 5.6, 5.7</td>
<td>Primary platform</td>
<td>Statement and row based replication is supported. MyISAM and InnoDB table types are fully supported; MyISAM tables may introduce replication errors during failover scenarios. JSON datatype support only available in Tungsten Replicator v5.3 onwards</td>
</tr>
<tr>
<td>Percona</td>
<td>5.5, 5.6, 5.7</td>
<td>Primary platform</td>
<td>Statement and row based replication is supported. MyISAM and InnoDB table types are fully supported; MyISAM tables may introduce replication errors during failover scenarios. JSON datatype support only available in Tungsten Replicator v5.3 onwards</td>
</tr>
<tr>
<td>MariaDB</td>
<td>5.5, 10.0, 10.1</td>
<td>Primary platform</td>
<td></td>
</tr>
</tbody>
</table>

---

### 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 (e.g. 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.
Prerequisites

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 Clustering 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 Clustering.
    ```
    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:
Prerequisites

- 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 Clustering™ 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 Clustering™ in a multi-master, multi-site, 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 Clustering™ 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 Clustering™ 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 7.14, “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 6.4, “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 master/slave 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 multi-master deployment, 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.
Prerequisites

- 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 slave.

- 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;
```

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 Multimaster Deployments**

If you are inserting to the same table at the same time at two or more different sites, and using bi-directional or multimaster 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:
Prerequisites

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 dataverse. 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.

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 Clustering 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 Clustering, and on the schemas that it will update when applying events.

The capability can be enabled by using the following two options and behaviors:
Prerequisites

• --privileged-master=false

When privileged master is disabled:

• A master replicator will not attempt to suppress binlog writes during operations.
• A master 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:

```sql
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

When privileged slave is disabled:

• The current replicator position is not updated within the trep_commit_seqno table.

```sql
mysql> GRANT ALL ON tungsten_batch.* to tungsten@'%' IDENTIFIED BY 'secret';
mysql> GRANT SELECT,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. Oracle Database Setup

Important

For Oracle extraction, Tungsten Clustering must have write access to the database so that status and progress information can be recorded correctly.

B.5.1. Oracle Version Support

<table>
<thead>
<tr>
<th>Database</th>
<th>Version</th>
<th>Support Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle</td>
<td>10g Release 2 (10.2.0.5), 11g</td>
<td>Primary Platform</td>
<td>Synchronous CDC is supported on Standard Edition only; Synchronous and Asynchronous are supported on Enterprise Editions.</td>
</tr>
</tbody>
</table>

B.5.2. Oracle Environment Variables

Ensure the tungsten user being used for the master Tungsten Replicator service has the same environment setup as an Oracle database user. The user must have the following environment variables set:

<table>
<thead>
<tr>
<th>Environment Variable</th>
<th>Sample Directory</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORACLE_HOME</td>
<td>/home/oracle/app/oracle/product/11.2.0/dbhome_2</td>
<td>The home directory of the Oracle installation.</td>
</tr>
<tr>
<td>LD_LIBRARY_PATH</td>
<td>$ORACLE_HOME/lib</td>
<td>The library directory of the Oracle installation.</td>
</tr>
<tr>
<td>ORACLE_SID</td>
<td>orcl</td>
<td>Oracle System ID for this installation.</td>
</tr>
<tr>
<td>JAVA_HOME</td>
<td></td>
<td>The home of the Java installation.</td>
</tr>
<tr>
<td>PATH</td>
<td>$ORACLE_HOME/bin:$JAVA_HOME/bin</td>
<td>Must include the Oracle and Java binary directories.</td>
</tr>
<tr>
<td>CLASSPATH</td>
<td>$ORACLE_HOME/lib/ucp/lib/ucp.jar:$ORACLE_HOME/jdbc/lib/ojdbc6.jar</td>
<td>Must include the key Oracle libraries the Oracle JDBC driver.</td>
</tr>
</tbody>
</table>

These should be set within the .bashrc or .profile to ensure these values are set correctly for all logins.

B.6. 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.
Prerequisites

A PDF version of this checklist can also be downloaded [here](#).

### Host Specific

<table>
<thead>
<tr>
<th>Pre-Req</th>
<th>Complete?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create OS User – Typically called <strong>tungsten</strong></td>
<td></td>
</tr>
<tr>
<td>Set <strong>ulimit</strong> for OS User</td>
<td></td>
</tr>
<tr>
<td>Configure sudoers</td>
<td></td>
</tr>
<tr>
<td>Disable SELinux</td>
<td></td>
</tr>
<tr>
<td>Compile <strong>/etc/hosts</strong></td>
<td></td>
</tr>
<tr>
<td>Setup SSH between hosts</td>
<td></td>
</tr>
<tr>
<td>Create directory for installation [Typically, <strong>/opt/continuent</strong>]</td>
<td></td>
</tr>
<tr>
<td>Create directory for software package if using tar bundle [Typically, <strong>/opt/continuent/software</strong>]</td>
<td></td>
</tr>
<tr>
<td>Create directory for configuration file if INI Install [<strong>/etc/tungsten</strong>]</td>
<td></td>
</tr>
<tr>
<td>Check ownership of new directories set to new OS user</td>
<td></td>
</tr>
<tr>
<td>Install Ruby</td>
<td></td>
</tr>
<tr>
<td>Install Ruby gems: <strong>net-ssh</strong></td>
<td></td>
</tr>
<tr>
<td>Install Ruby gems: <strong>net-scp</strong></td>
<td></td>
</tr>
<tr>
<td>Install Ruby gems: <strong>io-console</strong></td>
<td></td>
</tr>
<tr>
<td>Install Java 8</td>
<td></td>
</tr>
<tr>
<td>install rsync</td>
<td></td>
</tr>
</tbody>
</table>

### Network Specific

<table>
<thead>
<tr>
<th>Pre-Req</th>
<th>Complete?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ensure Network ports open</td>
<td></td>
</tr>
</tbody>
</table>

### Database Specific [All Topologies]

<table>
<thead>
<tr>
<th>Pre-Req</th>
<th>Complete?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ensure <strong>server-id</strong> unique amongst all nodes</td>
<td></td>
</tr>
<tr>
<td>Increase Open Files limits</td>
<td></td>
</tr>
<tr>
<td>Ensure <strong>bin-log</strong> enabled for cluster nodes, or source replicator nodes</td>
<td></td>
</tr>
<tr>
<td>Review <strong>sync_binlog</strong> parameter</td>
<td></td>
</tr>
<tr>
<td>Increase, if required, <strong>max_allowed_packet</strong></td>
<td></td>
</tr>
<tr>
<td>Review InnoDB settings</td>
<td></td>
</tr>
<tr>
<td>Set <strong>binlog_format</strong> to ROW (Essential for Multimaster or heterogeneous deployments)</td>
<td></td>
</tr>
<tr>
<td>Ensure <strong>auto_increment</strong> offsets adjusted for Multimaster deployments</td>
<td></td>
</tr>
<tr>
<td>Create DB user with FULL privileges and <strong>GRANT OPTION</strong> – typically called <strong>tungsten</strong> (Used by managers and repli-cators)</td>
<td></td>
</tr>
</tbody>
</table>
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.2, “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 [master, slave, 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
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.2, “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. 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
```

```shell
tungsten@host1:/home/tungsten/tungsten-replicator-5.4.1-41
```

```shell
cd /home/tungsten/tungsten-replicator-5.4.1-41
```

```shell
diag
```

```shell
tpm diag
```

480
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
   $ 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.1.4. 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.1.5. 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.1.6. 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.1.7. 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.1.8. 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 execute 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. |

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.

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.

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 uncomment two lines. For example:

```
shell> su - tungsten
shell> vi /opt/continuent/tungsten/tungsten-connector/conf/log4j.properties

Uncomment these two lines:

#log4j.logger.org.continuent.myosotis=debug, stdout
#log4j.additivity.org.continuent.myosotis=false

so they look like this:

log4j.logger.org.continuent.myosotis=debug, stdout
log4j.additivity.org.continuent.myosotis=false

shell> connector reconfigure
```

**Warning**

When disabling debug logging, DO NOT comment the lines out! Instead replace `debug` with `info`.

To disable debug logging, edit the Connector configuration file `tungsten-connector/conf/log4j.properties` and change the keyword `debug` to `info` for the single key `log4j.logger.org.continuent.myosotis`.

For example, this is how it should look when the edit to disable is completed:

```
shell> vi /opt/continuent/tungsten/tungsten-connector/conf/log4j.properties
log4j.logger.org.continuent.myosotis=info, stdout
...
shell> connector reconfigure
```

## Configuring Connector Verbose Debug Logging

This procedure allows the Connector to be configured for verbose debug logging (and we mean VERBOSE).

Perform this procedure on Connector hosts when requested by Continuent support.

**Warning**

Enabling Connector verbose 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 verbose debug logging (and we mean VERBOSE), edit the Connector log configuration file `tungsten-connector/conf/log4j.properties` and ADD two lines. For example:

```
shell> su - tungsten
shell> vi /opt/continuent/tungsten/tungsten-connector/conf/log4j.properties

log4j.logger.org.continuent.myosotis.protocol.mysql.MySQLProtocolHandler=trace, stdout
log4j.additivity.org.continuent.myosotis.protocol.mysql.MySQLProtocolHandler=false

shell> connector reconfigure
```

**Warning**

When disabling verbose debug logging, DO NOT comment the lines out! Instead replace `trace` with `info`.

To disable verbose debug logging, edit the Connector log configuration file `tungsten-connector/conf/log4j.properties` and change the keyword `trace` to `info` for the single key `log4j.logger.org.continuent.myosotis.protocol.mysql.MySQLProtocolHandler`.

For example, this is how it should look when the edit to disable is completed:

```
shell> vi /opt/continuent/tungsten/tungsten-connector/conf/log4j.properties
...
log4j.logger.org.continuent.myosotis.protocol.mysql.MySQLProtocolHandler=info, stdout
...
shell> connector reconfigure
```

## Configuring Connector Debug Logging via the Manager

This procedure allows the Manager to be configured to enable Connector-specific debug logging in the manager logs.
Troubleshooting

Perform this procedure on Manager hosts when requested by Continuent support.

**Warning**
Enabling Manager-based Connector debug logging will decrease performance. 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 Manager-based Connector debug logging, edit the Manager log configuration file `tungsten-manager/conf/log4j.properties` and add two lines. For example:

```shell
su - tungsten
vi /opt/continuent/tungsten/tungsten-manager/conf/log4j.properties
log4j.logger.com.continuent.tungsten.manager.router.gateway.RouterGatewayProxy=DEBUG, stdout
log4j.additivity.com.continuent.tungsten.manager.router.gateway.RouterGatewayProxy=false

manager restart
```

**Warning**
When disabling Manager-based Connector debug logging, DO NOT comment the lines out! Instead replace `DEBUG` with `info`.

To disable Manager-based Connector debug logging, edit the Manager log configuration file `tungsten-manager/conf/log4j.properties` and change the keyword `DEBUG` to `info` for the single key `log4j.logger.com.continuent.tungsten.manager.router.gateway.RouterGatewayProxy`.

For example, this is how it should look when the edit to disable is completed:

```shell
vi /opt/continuent/tungsten/tungsten-manager/conf/log4j.properties
log4j.logger.com.continuent.tungsten.manager.router.gateway.RouterGatewayProxy=info, stdout

manager restart
```

C.2. Error/Cause/Solution

C.2.1. 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 10.4.30, “PrimaryKey Filter”

C.2.2. 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.
Troubleshooting

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 \([\text{replicator.global.buffer.size}]\) 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:

\[
\text{replicator.global.buffer.size}=10
\]

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 over the network to slave replicators. Reducing the fragment size will reduce the memory footprint. The size is controlled by the \text{replicator.extractor.dbms.transaction_frag_size} parameter:

\[
\text{replicator.extractor.dbms.transaction_frag_size}=1000000
\]

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 \text{trepctl reset} command.

C.2.3. Unable to update the configuration of an installed directory

Last Updated: 2013-08-07

Condition or Error

Running an update or configuration with \text{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 Clustering was originally installed.

Rectifications

- Change to the staging directory and perform the necessary commands with \text{tpm}. To determine the staging directory, use:

\[
\text{shell> tpm query staging}
\]

Then change to the staging directory and perform the updates:

\[
\text{shell> ./tools/tpm configure ....}
\]

More Information

Chapter 2, Deployment Overview

C.2.4. 'subscription exists' when setting up CDC on Oracle

Last Updated: 2014-04-24

Condition or Error

When running \text{setupCDC.sh} an error regarding an existing subscription is received and \text{setupCDC.sh} fails to complete.

Causes

- This error indicates that \text{setupCDC.sh} has previously been executed and the CDC configuration and publishers and subscribers have been created, but the names of the tables or users may have been changed.

Rectifications

- The \text{cleanup_cdc_tables.sql} file can be used to cleanup an existing CDC installation. See Section 4.1.7, "CDC Cleanup and Correction".
C.2.5. 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 Clustering 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.2.6. 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(): 1 threads didn't exit`

Rectifications

• To resolve this issue in Centos, install openssl by running the command:
C.2.7. 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:

```sql
mysql> ALTER TABLE faulty MODIFY COLUMN faulty_column DECIMAL;
```

This should be performed on the master within your topology. To correct the error, you must use tpm reset-thl to regenerate the THL.

C.2.8. 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/slave from the new master 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 slave offline:
   ```
   slave-shell> trepctl offline
   ```

2. Take the master offline:
   ```
   slave-shell> trepctl offline
   ```

3. Use trepctl to reset the service on the master and slave. You must use the service name explicitly on the command-line:
   ```
   master-shell> trepctl -service alpha reset -y
   slave-shell> trepctl -service alpha reset -y
   ```

4. Put the slave online:
   ```
   slave-shell> trepctl offline
   ```
5. Put the master online:

```
slave-shell> trepctl offline
```

**C.2.9. ORA-00257: ARCHIVER ERROR. CONNECT INTERNAL ONLY, UNTIL FREED**

*Last Updated: 2016-04-20*

**Condition or Error**

It is possible for the Oracle server to get into a state where Tungsten Clustering is online, and with no other errors showing in the log. However, when logging into the Oracle server an error is returned:

```
ORA-00257: ARCHIVER ERROR. CONNECT INTERNAL ONLY, UNTIL FREED
```

**Causes**

- This is a lack of resources within the Oracle server, and not an issue with Tungsten Clustering.

**Rectifications**

- The issue can be addressed by increasing the logical size of the recovery area, by connecting to the Oracle database as the system user and running the following command:

```
shell> sqlplus sys/oracle as sysdba
SQL> ALTER SYSTEM SET db_recovery_file_dest_size = 80G;
```

**C.2.10. 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.2.11. The session variable SQL_MODE when set to include ALLOW_INVALID_DATES does not apply statements correctly on the slave.**

*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.
Troubleshooting

- 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. Known Issues

C.3.1. Triggers

Tungsten Replicator does not automatically shut off triggers on slaves. This can create problems on slaves 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 Trigger 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_master()` 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 slave 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_master()` 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), 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), or
  - Switch to **ROW** Based Binary Logging and follow recommendations above

The `is_master()` 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 slave.

```sql
CREATE function is_master()
returns boolean
deterministic
return if(substring_index(user(),'@', 1) != 'tungsten', true, false);
```

Next add this to triggers that should not run on the slave, as shown in the next example. This suppresses trigger action to insert into table bar except on the master.

```sql
delimiter //
create trigger insert_bar_row
before insert on bar
for each row
begin
  set new.bar = new.bar + 1;
end
//
```

489
create trigger foo_insert after insert on foo
for each row begin
  if is_master() then
    insert into bar set id=NEW.id;
  end if;
end;
//

As long as applications do not use the Tungsten account on the master, the preceding approach will be sufficient to suppress trigger operation.

Alternatively, if you are implementing the \texttt{is\_master()} within a clustering deployment, you could check the database \texttt{read\_only} parameter. In a clustered deployment, the slave databases will be in \texttt{read\_only} mode and therefore the trigger could be coded to only fire when the database \texttt{read\_only} mode is \texttt{OFF}.

C.4. Troubleshooting Timeouts

C.5. Troubleshooting Backups

- Operating system command failed

Backup directory does not exist.

The directory '/opt/continuent/backups/xtrabackup' is not writeable.

- Backup Retention

C.6. Running Out of Diskspace

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.7. Troubleshooting SSH and tpm

When executing \texttt{tpm}, \texttt{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 \texttt{tpm} during a deployment:

Try running the following command:
Troubleshooting

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 [334] option to specify the alternative port.

- Make sure that the user specified in the --user to tpm is allowed to connect to your cluster nodes.

C.8. 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.8.1. 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 Master node, specifying the schema in place of {DB}:

```
shell> mysqldump -u root -p --no-data -h localhost --databases {DB} >master.sql
```

Repeat the same on the Slave node:

```
shell> mysqldump -u root -p --no-data -h localhost --databases {DB} >slave.sql
```

Now, using diff, you can compare the results

```
shell> diff master.sql slave.sql
```

Using the output of diff, you can then craft the necessary SQL statements to re-align your structure

C.8.2. 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:


Once you are ready, ensure you install the latest version to the persona toolkit on all nodes, next execute the following on the Master 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 slave to identify tables with data differences:

```sql
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 \texttt{--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 \texttt{tungsten\_provision\_slave} to retrovision a node in the case of large quantities of differences.

To use \texttt{pt-table-sync}, first identify the tables with differences on each slave, in this example, the \texttt{SELECT} statement above identified that there was a data difference on the departments table within the employees database on \texttt{db2}. Execute the \texttt{pt-table-sync} script on the master, passing in the database name, table name and the slave 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 \texttt{h=} option should be the Master, also the node you run the script from, the second \texttt{h=} option relates to the slave 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 \texttt{UPDATE} statements could now be issued directly on the slave to correct the problem.

**Warning**

Generally, changing data directly on a slave 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 slave node (if in a clustered environment) before making any changes so as not to cause any potential interruption to connected clients.

**C.9. Comparing Table Data**

The Percona Toolkit includes a tool called \texttt{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 master:

```
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 master:

```
mysql> \texttt{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;\texttt{userinput}
```

Any differences will be reported and will need to manually corrected.

**C.10. Troubleshooting Memory Usage**
Appendix D. Files, Directories, and Environment

D.1. The Tungsten Clustering Install Directory

Any Tungsten Clustering™ 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”.

```
ls -al /opt/continuent
```

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

<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>backups</td>
<td>Default directory for backup file storage</td>
</tr>
<tr>
<td>conf</td>
<td>Configuration directory with a copy of the current and past configurations</td>
</tr>
<tr>
<td>relay</td>
<td>Location for relay logs if relay logs have been enabled.</td>
</tr>
<tr>
<td>releases</td>
<td>Contains one or more active installations of the Continuent Tungsten software, referenced according to the version number and active process ID.</td>
</tr>
<tr>
<td>service-logs</td>
<td>Logging information for the active installation</td>
</tr>
<tr>
<td>share</td>
<td>Active installation information, including the active JAR for the MySQL connection</td>
</tr>
<tr>
<td>thl</td>
<td>The Transaction History Log files, stored in a directory named after each active service.</td>
</tr>
<tr>
<td>tungsten</td>
<td>Symbolic link to the currently active release in releases.</td>
</tr>
</tbody>
</table>

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 Clustering™. The backup data and metadata for each backup will be stored in this directory.

An example of the directory content is shown below:

```
ls -al /opt/continuent/backups/
```

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 info /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 [366] 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 [366].

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
Drwxr-xr-x  2 tungsten tungsten      4096 Apr 16 13:57 .
Drwxr-xr-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-228485797798800458.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-[0-9]*@("...") 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 -ae 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_58
Drwxr-xr-x  7 tungsten tungsten 4096 Oct 19 20:55 incr_xtrabackup_2014-10-19_20-57_76
```
Files, Directories, and Environment

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
  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:
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.1.4.2. 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` 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 28
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-5.4.1-41_pid24978/
drwxr-xr-x 10 tungsten mysql 4096 May 23 16:19 tungsten-replicator-5.4.1-41_pid24978/
drwxr-xr-x 10 tungsten mysql 4096 May 23 16:19 tungsten-replicator-5.4.1-41_pid24978/
drwxr-xr-x 10 tungsten mysql 4096 May 23 16:19 tungsten-replicator-5.4.1-41_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 23 16:19 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  2 tungsten mysql 4096 May 10 19:09 thl/
lrwxrwxrwx  1 tungsten mysql   63 May 23 16:19 tungsten -> /opt/continuent/releases/tungsten-replicator-5.4.1-41_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-5.4.1-41_pid24978/tungsten-replicator/var/treplicator.pid
/opt/continuent/releases/tungsten-replicator-5.4.1-41_pid24978/tungsten-connector/var/lconnector.pid
/opt/continuent/releases/tungsten-replicator-5.4.1-41_pid24978/tungsten-manager/var/manager.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:

- `trepvsc.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 Clustering. 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 master 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 18:20 thl.data.0000000003
-rw-r--r-- 1 tungsten tungsten 100287700 Apr 15 18:23 thl.data.0000000004
-rw-r--r-- 1 tungsten tungsten 100438994 Apr 15 18:34 thl.data.0000000005
-rw-r--r-- 1 tungsten tungsten 100397260 Apr 15 18:35 thl.data.0000000006
-rw-r--r-- 1 tungsten tungsten 100294561 Apr 16 12:03 thl.data.0000000007
-rw-r--r-- 1 tungsten tungsten 100332338 Apr 16 12:24 thl.data.0000000008
-rw-r--r-- 1 tungsten tungsten 100218972 Apr 16 12:35 thl.data.0000000009
-rw-r--r-- 1 tungsten tungsten 10037285 Apr 16 12:37 thl.data.0000000010
-rw-r--r-- 1 tungsten tungsten 10053387 Apr 16 12:38 thl.data.0000000011
-rw-r--r-- 1 tungsten tungsten 10073318 Apr 16 12:48 thl.data.0000000012
-rw-r--r-- 1 tungsten tungsten 10098421 Apr 16 13:32 thl.data.0000000013
-rw-r--r-- 1 tungsten tungsten 100136955 Apr 16 13:34 thl.data.0000000014
-rw-r--r-- 1 tungsten tungsten 10049927 Apr 16 13:41 thl.data.0000000015
-rw-r--r-- 1 tungsten tungsten 10064146 Apr 16 13:41 thl.data.0000000016
-rw-r--r-- 1 tungsten tungsten 100225119 Apr 16 13:42 thl.data.0000000017
-rw-r--r-- 1 tungsten tungsten 10039819 Apr 16 13:43 thl.data.0000000018
-rw-r--r-- 1 tungsten tungsten 10048115 Apr 16 13:44 thl.data.0000000019
-rw-r--r-- 1 tungsten tungsten 10038812 Apr 16 13:44 thl.data.0000000020
-rw-r--r-- 1 tungsten tungsten 10018759 Apr 16 13:47 thl.data.0000000021
```

THL files are created on both the master and slaves 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(1512:1551)
LogIndexEntry thl.data.0000000012(1551:1699)
LogIndexEntry thl.data.0000000013(1699:1700)
LogIndexEntry thl.data.0000000014(1701:1800)
LogIndexEntry thl.data.0000000015(1801:1900)
LogIndexEntry thl.data.0000000016(1901:1997)
LogIndexEntry thl.data.0000000017(1998:2493)
LogIndexEntry thl.data.0000000018(2494:2598)
LogIndexEntry thl.data.0000000019(2599:2754)
LogIndexEntry thl.data.0000000020(2755:2851)
LogIndexEntry thl.data.0000000021(2852:2948)
LogIndexEntry thl.data.0000000022(2949:3045)
LogIndexEntry thl.data.0000000023(3046:3142)
LogIndexEntry thl.data.0000000024(3143:3239)
```

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 Slave

Warning

Purging the THL on a slave 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 number to the database is not purged, and that additional hosts are not reading the THL information.

To purge the logs on a SLAVE 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 8.17.4, “thl purge Command”.

D.1.5.2. Purging THL Log Information on a Master

Warning

Purging the THL on a Master node can potentially remove information that has not yet been applied to the slave databases. Please check and ensure that the THL data that you are purging has been applied to the database on all slaves before continuing.

Important

If the situation allows, it may be better to switch the Master role to a current, up-to-date slave, then perform the steps to purge THL from a slave on the old master host using Section D.1.5.1, “Purging THL Log Information on a Slave”.

Warning

Follow the below steps with great caution! Failure to follow best practices will result in slaves unable to apply transactions, forcing a full re-provisioning. For those steps, please see Section 7.6, “Provision or Reprovision a Slave”.

The THL files can be explicitly purged to recover disk space, but you should ensure that the currently applied sequence number to the database is not purged, and that additional hosts are not reading the THL information.
To purge the logs on a MASTER 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 8.17.4, “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”.

#### 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 slave by temporarily setting the slave 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.1.3.2. Relocating THL Storage using Configuration Changes

To permanently change the directory currently holding the THL information can be reconfigured to a new directory location.

To update the location for a slave by temporarily setting the slave 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 to the 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 Clustering. 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` 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**

<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bristlecone</td>
<td>Contains the bristlecone load-testing tools.</td>
</tr>
<tr>
<td>cluster-home</td>
<td>Home directory for the main tools, configuration and libraries of the Tungsten Clustering installation.</td>
</tr>
<tr>
<td>cookbook</td>
<td>Cookbook installation and testing tools.</td>
</tr>
</tbody>
</table>
Directory | Description
--- | ---
INSTALL | Text file describing the basic installation process for Tungsten Clustering
README.LICENSES | Software license information.
tools | Directory containing the tools for installing and configuring Tungsten Clustering.
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 product.

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 its 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
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, along with logs for backups, if they exist.

The more detailed log files can be found in each of the "real" log directories, for example:

```
/opt/continuent/tungsten/tungsten-replicator/log/trepsvc.log
```

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 Clustering. When defined, the variable will contain the installation directory of the corresponding Tungsten Clustering installation.

  On hosts where multiple installations have been created, the variable can be used to point to different installations.
Appendix E. Terminology Reference

Tungsten Clustering 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 Clustering.

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 master, 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 be 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-r--r-- 1 tungsten tungsten 101835311 Jun  4 11:41 thl.data.0000000014
-rw-r--r-- 1 tungsten tungsten 100441159 Jun  4 11:41 thl.data.0000000015
-rw-r--r-- 1 tungsten tungsten 100898492 Jun  4 11:44 thl.data.0000000016
-rw-r--r-- 1 tungsten tungsten 100356113 Jun  4 11:44 thl.data.0000000017
-rw-r--r-- 1 tungsten tungsten 100685516 Jun  4 11:44 thl.data.0000000018
-rw-r--r-- 1 tungsten tungsten 101689869 Jun  4 11:45 thl.data.0000000019
-rw-r--r-- 1 tungsten tungsten 23886641 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` 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` configuration parameter. The files can be manually purged or moved. See Section D.1.5.1, “Purging THL Log Information on a Slave”.

The THL can be viewed and managed by using the `thl` command. For more information, see Section 8.17, “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=dson1;]```
The sample above shows the information for the SQL executed on a MySQL server. The `eventid` [504] 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>
<thead>
<tr>
<th>Displayed Field</th>
<th>Internal Name</th>
<th>Data type</th>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>record_length</td>
<td>Integer</td>
<td>4 bytes</td>
<td></td>
<td>Length of the full record information, including this field</td>
</tr>
<tr>
<td>record_type</td>
<td>Byte</td>
<td>1 byte</td>
<td></td>
<td>Event record type identifier</td>
</tr>
<tr>
<td>header_length</td>
<td>Unsigned int</td>
<td>4 bytes</td>
<td></td>
<td>Length of the header information</td>
</tr>
<tr>
<td>seqno</td>
<td>Unsigned long</td>
<td>8 bytes</td>
<td></td>
<td>Log sequence number, a sequential value given to each log entry</td>
</tr>
<tr>
<td>fragno</td>
<td>Unsigned short</td>
<td>2 bytes</td>
<td></td>
<td>Event fragment number. An event can consist of multiple fragments of SQL or row log data</td>
</tr>
<tr>
<td>last_frag</td>
<td>Byte</td>
<td>1 byte</td>
<td></td>
<td>Indicates whether the fragment is the last fragment in the sequence</td>
</tr>
<tr>
<td>epoch_number</td>
<td>Unsigned long</td>
<td>8 bytes</td>
<td></td>
<td>Event epoch number. Used to identify log sections within the master THL</td>
</tr>
<tr>
<td>source_id</td>
<td>UTF-8 String</td>
<td>Variable</td>
<td></td>
<td>Event source ID, the hostname or identity of the datasource that generated the event</td>
</tr>
<tr>
<td>event_id</td>
<td>UTF-8 String</td>
<td>Variable</td>
<td></td>
<td>Event ID; in MySQL, for example, the binlog file-name and position that contained the original event</td>
</tr>
<tr>
<td>shard_id</td>
<td>UTF-8 String</td>
<td>Variable</td>
<td></td>
<td>Shard ID to which the event belongs</td>
</tr>
<tr>
<td>tstamp</td>
<td>Unsigned long</td>
<td>8 bytes</td>
<td></td>
<td>Time of the commit that triggered the event</td>
</tr>
<tr>
<td>data_length</td>
<td>Unsigned int</td>
<td>4 bytes</td>
<td></td>
<td>Length of the included event data</td>
</tr>
<tr>
<td>event</td>
<td>Binary</td>
<td>Variable</td>
<td></td>
<td>Serialized Java object containing the SQL or ROW data</td>
</tr>
<tr>
<td>metadata</td>
<td>Part of event</td>
<td>-</td>
<td>-</td>
<td>Metadata about the event</td>
</tr>
<tr>
<td>type</td>
<td>Part of event</td>
<td>-</td>
<td>-</td>
<td>Internal storage type of the event</td>
</tr>
<tr>
<td>options</td>
<td>Part of event</td>
<td>-</td>
<td>-</td>
<td>Options about the event operation</td>
</tr>
<tr>
<td>schema</td>
<td>Part of event</td>
<td>-</td>
<td>-</td>
<td>Schema used in the event</td>
</tr>
<tr>
<td>sql</td>
<td>Part of event</td>
<td>-</td>
<td>-</td>
<td>SQL statement or row data</td>
</tr>
<tr>
<td>crc_method</td>
<td>Byte</td>
<td>1 byte</td>
<td></td>
<td>Method used to compute the CRC for the event.</td>
</tr>
<tr>
<td>crc</td>
<td>Unsigned int</td>
<td>4 bytes</td>
<td></td>
<td>CRC of the event record (not including the CRC value)</td>
</tr>
</tbody>
</table>

- `seqno` [503] and `fragno` [503]

Individual events within the log are identified by a sequential `sequence` [503] 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 slaves and master are in sync, and can be used to identify individual transactions within the replication stream.
• **EPOCH [504]**

The **EPOCH [504]** value is used as a check to ensure that the logs on the slave and the master match. The **EPOCH [504]** is stored in the THL, and a new **EPOCH [504]** is generated each time a master goes online. The **EPOCH [504]** value is then written and stored in the THL alongside each individual event. The **EPOCH [504]** acts as an additional check, beyond the sequence number, to validate the information between the slave and the master. The **EPOCH [504]** value is used to prevent the following situations:

- In the event of a failover where there are events stored in the master log, but which did not make it to a slave, the **EPOCH [504]** acts as a check so that when the master rejoins as the slave, the **EPOCH [504]** numbers will not match the slave and the new master. The trapped transactions be identified by examining the THL output.

- When a slave joins a master, the existence of the **EPOCH [504]** prevents the slave from accepting events that happen to match only the sequence number, but not the corresponding **EPOCH [504]**.

Each time a Tungsten Replicator master goes online, the **EPOCH [504]** number is incremented. When the slave connects, it requests the **SEQUENCE [503]** and **EPOCH [504]**, and the master confirms that the requested **SEQUENCE [503]** has the requested **EPOCH [504]**. If not, the request is rejected and the slave 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 master and slave to determine if there really is a mismatch between the two databases. For more information, see Section 7.5, “Managing Transaction Failures”.

• **SOURCEID [504]**

The **SOURCEID [504]** is a string identifying the source of the event stored in the THL. Typically it is the hostname or host identifier.

• **EVENTID [504]**

The **EVENTID [504]** is a string identifying the source of the event information in the log. Within a MySQL installed, the **EVENTID [504]** 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 [504]**

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 [504]** value as stored in the THL is used to compute latency information when reading and applying data on a slave.

• **METADATA [504]**

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 master 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 [504]**

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 [504]**
Part of the EVENT binary payload, the OPTIONS [504] 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 [504] block for each SQL [505] statement stored in the event.

- **SCHEMA [505]**

  Part of the EVENT structure, the SCHEMA [505] provides the database or schema name in which the statement or row data was applied.

- **SHARDID [505]**

  When using parallel apply, provides the generated shard ID for the event when it is applied by the parallel applier thread. data.

- **SQL [505]**

  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 [505], TABLE [505] and individual ROW data. For each ROW, there may be one or more COL [505] (column) and identifying KEY [505] event to identify the row on which the action is to be performed.

The same statement when recorded in ROW [505] 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 Clustering 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 `accessFailures`

#### E.2.2. Terminology: Fields `active`
E.2.3. Terminology: Fields activeSeqno

E.2.4. 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>trepcctl 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:36626156
...```

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 : 893187179244008818.002.00019601#LAST#8931871792337W180644
```

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.2.5. 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 slave database.

```
appliedLastSeqno : 212```

When using parallel replication, this parameter returns the minimum applied sequence number among all the channels applying data.

E.2.6. 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.
Terminology Reference

Within a master, this indicates the latency between the transaction commit time and when it was written to the THL. In a slave, it indicates the latency between the commit time on the master 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 measured in seconds. Increasing latency may indicate that the destination database is unable to keep up with the transactions from the master.

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.2.7. Terminology: Fields `applier.class`
Classname of the current applier engine

E.2.8. Terminology: Fields `applier.name`
Name of the current applier engine

E.2.9. Terminology: Fields `applyTime`

E.2.10. Terminology: Fields `autoRecoveryEnabled`
Indicates whether autorecovery has been enabled by setting the `--auto-recovery-max-attempts` [365]. The field indicates the value as either `true` or `false` accordingly.

E.2.11. 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 `trepvc.log` file.

The counter is reset when the replicator determines that the replicator has successfully gone online after an autorecovery.

E.2.12. Terminology: Fields `averageBlockSize`

E.2.13. Terminology: Fields `blockCommitRowCount`


E.2.15. Terminology: Fields `channel`

E.2.16. 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.2.17. 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.2.18. Terminology: Fields `commits`

E.2.19. Terminology: Fields `committedMinSeqno`
E.2.20. Terminology: Fields criticalPartition

E.2.21. Terminology: Fields currentBlockSize

E.2.22. Terminology: Fields currentEventId
    Event ID of the transaction currently being processed

E.2.23. Terminology: Fields currentLastEventId

E.2.24. Terminology: Fields currentLastFragno

E.2.25. Terminology: Fields currentLastSeqno

    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.2.27. Terminology: Fields dataServerHost

E.2.28. Terminology: Fields discardCount

E.2.29. Terminology: Fields doChecksum

E.2.30. Terminology: Fields estimatedOfflineInterval

E.2.31. Terminology: Fields eventCount

E.2.32. Terminology: Fields extensions

E.2.33. Terminology: Fields extractTime

E.2.34. Terminology: Fields extractor.class

E.2.35. Terminology: Fields extractor.name

E.2.36. Terminology: Fields filter.\$.class

E.2.37. Terminology: Fields filter.\$.name

E.2.38. Terminology: Fields filterTime
E.2.39. Terminology: Fields *flushIntervalMillis*

E.2.40. Terminology: Fields *fsyncOnFlush*

E.2.41. Terminology: Fields *headSeqno*

E.2.42. Terminology: Fields *intervalGuard*

E.2.43. 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 11.1, "Block Commit".

E.2.44. Terminology: Fields *lastCommittedBlockTime*

The *lastCommittedBlockTime* 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 11.1, "Block Commit".

E.2.45. Terminology: Fields *latestEpochNumber*

E.2.46. Terminology: Fields *logConnectionTimeout*

E.2.47. Terminology: Fields *logDir*

E.2.48. Terminology: Fields *logFileSize*

E.2.49. Terminology: Fields *logFileRetainMillis*

E.2.50. Terminology: Fields *masterConnectUri*

The URI being used to extract THL information. On a master, the information may be empty, or may contain the reference to the underlying extractor source where information is being read.

On a slave, 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.2.51. Terminology: Fields *masterListenUri*

The URI on which the replicator is listening for incoming slave requests. On a master, this is the URI used to distribute THL information.

```
masterListenUri : thl://host1:2112/
```

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  
**maxStoredSeqNo**

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.

| maxStoredSeqNo | 25 |

E.2.57. Terminology: Fields  
**minStoredSeqNo**

The minimum transaction ID stored locally in the THL on the host:

| minStoredSeqNo | 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
... minimumStoredSeqNo : 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 master, the pipeline source is the database that the master is connected to and extracting data from. Within a slave, the pipeline source is the master 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 master, it indicates whether the master is keeping up with transactions generated on the master database. On a slave, it indicates how up to date the THL read from the master 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 master replicator, or that the replicator has stalled for some reason.

An increasing relativeLatency on the slave 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 Clustering 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 that are designated as internal features and functionality.

This chapter contains information on the following sections of Tungsten Clustering:

- 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 Clustering” — 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 Clustering.

F.1. Extending Backup and Restore Behavior

The backup and restore system within Tungsten Clustering 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 Clustering 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 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:

```bash
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 script 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:

```ruby
#!/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
    # Write the filename to the final storage file
    TU.cmd_result("echo "file=#{storage_file}" > #{storage_file}")
    # Write the filename to the final storage file
  end
end
```

The properties file used by Tungsten Replicator to exchange information about the backup or restore.

```bash
-properties FILENAME
```

The properties file is used by Tungsten Replicator to exchange information about the backup or restore.

```bash
-options ARG1=VAL1&ARG2=VAL2
```

The script has to determine the files that were generated during the backup process and collect them into a single file as appropriate.

```bash
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.
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` [366] should be set to `script`, and the directory location set using the `-repl-backup-script` [367] 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` [366] 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 \textit{-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:

\begin{verbatim}
shell: trepctl -host host2 backup -backup script
\end{verbatim}

\textbf{Note}

Note that the name of the backup method is \textit{script}, not the actual name of the script being used.

\section*{F.2. Character Sets in Database and Tungsten Clustering}

Character sets within the databases and within the configuration for Java and the wrappers for Tungsten Clustering 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 Clustering, be aware of the following aspects when using character sets:

\begin{itemize}
  \item When installing Tungsten Clustering, use the \texttt{-java-file-encoding} to configure the character set.
  \item When using the \texttt{thl} command, the character set may need to be explicitly stated to view the content correctly:
    \begin{verbatim}
    shell: thl list -charset utf8
    \end{verbatim}
\end{itemize}

For more information on setting character sets within your database, see your documentation for the database:

\begin{itemize}
  \item MySQL
  \item Oracle
\end{itemize}

For more information on the character set names and support within Java, see:

\begin{itemize}
  \item Java 6 SE
  \item Java 7 SE
\end{itemize}

\section*{F.3. Understanding Replication of Date/Time Values}

\begin{itemize}
  \item Replicator processes default to UTC internally by setting the Java VM default time zone to UTC. This default can be changed by setting the \texttt{replicator.time\_zone} property in the replicator services.properties file but is not recommended other than for problem diagnosis or specialized testing.
  \item Replicas store a time zone on statements and row changes extracted from MySQL.
  \item Replicators use UTC as the session time zone when applying to MySQL replicas.
  \item Replicators similarly default to UTC when applying transactions to data warehouses like Hadoop, Vertica, or Amazon Redshift.
  \item The \texttt{thl} utility prints time-related data using the default GMT time zone. This can be altered using the \texttt{-timezone} option.
\end{itemize}

\section*{Best Practices}

We recommend the following steps to ensure successful replication of time-related data:

\begin{itemize}
  \item 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.
  \item Use the default time zone settings for Tungsten replicator. Do not change the time zones unless specifically recommended by Continuent support.
  \item If you cannot standardize on UTC at least ensure that time zones are set consistently on all hosts and applications.
\end{itemize}

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 Clustering 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 replicas run rarely 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 see from the attached diagram, a slave pipeline might have a stage to read transactions to the master 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 slave. 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 slaves 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` 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` parameter at installation time.)

- **Property repliator.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` 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 slave 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 slaves 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 bin-log 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 master, and the local THL on disk for a slave
- **Filter**: any configured filters are applied here
- **Apply**: the apply target can be THL on disk on a master, and the database server on a slave

Stages can be customized with filters, and filters are invoked on a per-stage basis.

By default, there are two pipeline services defined:

- **Master 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.
- **Slave replication service**, which contains three (3) stages:
  - **remote-to-thl**: remote THL information is read from a master 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 Clustering Schemas**
Appendix G. Frequently Asked Questions (FAQ)

G.1. One of my hosts is regularly a number of seconds behind my other slaves?

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 master to slaves and checking the latency:

```
shell> trepctl heartbeat
```

G.2. 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.3. 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
```

G.4. On a Tungsten Replicator slave, how do I set both the local slave THL listener port and the upstream master’s THL listener port?

You need to specify two options: thl-port [403] to set the slave THL listener port and master-thl-port [386] to define the upstream master THL listener port. Otherwise thl-port [403] alone sets BOTH.
Appendix H. Ecosystem Support

In addition to the core utilities provided by Tungsten Clustering, 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 Clustering 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:

```
shell> git clone https://github.com/continuent/continuent-tools-hadoop.git
```

This will create a clone of the repository within the current directory.

To keep the copy up to date with any changes in the main Github repository:

```
shell> cd continuent-tools-hadoop
shell> git pull
```

The following tools and functionality are available within these repositories:

- **continuent-tools-hadoop**

  Provides tools to support the processing and materialization of view data within Hadoop. The repository contains two primary tools, `load-reduce-check` and `materialize`. 
Appendix I. Configuration Property Reference