Abstract

This manual documents Continuent Tungsten 2.0, up to and including 2.0.5.

Build date: 2015-04-14, Revision: 1282

Up to date builds of this document: Continuent Tungsten 2.0 Manual (Online), Continuent Tungsten 2.0 Manual (PDF)
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preface</td>
<td>xiv</td>
</tr>
<tr>
<td>1. Legal Notice</td>
<td>xiv</td>
</tr>
<tr>
<td>2. Conventions</td>
<td>xv</td>
</tr>
<tr>
<td>1. Introduction</td>
<td>16</td>
</tr>
<tr>
<td>1.1. Tungsten Replicator</td>
<td>16</td>
</tr>
<tr>
<td>1.1.1. Transaction History Log (THL)</td>
<td>17</td>
</tr>
<tr>
<td>1.2. Tungsten Manager</td>
<td>17</td>
</tr>
<tr>
<td>1.3. Tungsten Connector</td>
<td>17</td>
</tr>
<tr>
<td>1.4. Key Terminology</td>
<td>18</td>
</tr>
<tr>
<td>1.5. Datasource Types</td>
<td>18</td>
</tr>
<tr>
<td>2. Deployment</td>
<td>19</td>
</tr>
<tr>
<td>2.1. Host Types</td>
<td>19</td>
</tr>
<tr>
<td>2.1.1. Manager Hosts</td>
<td>19</td>
</tr>
<tr>
<td>2.1.2. Connector (Router) Hosts</td>
<td>20</td>
</tr>
<tr>
<td>2.1.3. Replicator Hosts</td>
<td>20</td>
</tr>
<tr>
<td>2.1.4. Witness Hosts</td>
<td>20</td>
</tr>
<tr>
<td>2.2. Requirements</td>
<td>22</td>
</tr>
<tr>
<td>2.2.1. Operating Systems Support</td>
<td>22</td>
</tr>
<tr>
<td>2.2.2. Database Support</td>
<td>22</td>
</tr>
<tr>
<td>2.2.3. RAM Requirements</td>
<td>23</td>
</tr>
<tr>
<td>2.2.4. Disk Requirements</td>
<td>23</td>
</tr>
<tr>
<td>2.2.5. Java Requirements</td>
<td>23</td>
</tr>
<tr>
<td>2.2.6. Cloud Deployment Requirements</td>
<td>23</td>
</tr>
<tr>
<td>2.3. Deployment Sources</td>
<td>24</td>
</tr>
<tr>
<td>2.3.1. Using the TAR/GZipped files</td>
<td>25</td>
</tr>
<tr>
<td>2.3.2. Using the RPM and DEB package files</td>
<td>25</td>
</tr>
<tr>
<td>2.4. Deploying a Master/Slave Topology</td>
<td>26</td>
</tr>
<tr>
<td>2.4.1. Installing a Master/Slave Configuration</td>
<td>26</td>
</tr>
<tr>
<td>2.5. Deploying a Multisite/Multimaster Topology</td>
<td>27</td>
</tr>
<tr>
<td>2.5.1. Preparing Hosts for Multimaster</td>
<td>27</td>
</tr>
<tr>
<td>2.5.2. Installing Multisite, Multimaster Topologies</td>
<td>30</td>
</tr>
<tr>
<td>2.5.3. Management and Monitoring</td>
<td>34</td>
</tr>
<tr>
<td>2.5.3.1. Configuring Startup on Boot</td>
<td>35</td>
</tr>
<tr>
<td>2.5.3.2. Resetting a single dataservice</td>
<td>36</td>
</tr>
<tr>
<td>2.5.3.3. Resetting all dataservices</td>
<td>36</td>
</tr>
<tr>
<td>2.5.3.4. Provisioning during live operations</td>
<td>36</td>
</tr>
<tr>
<td>2.5.3.5. Dataserver maintenance</td>
<td>37</td>
</tr>
<tr>
<td>2.6. Deploying Composite (SOR) Topologies</td>
<td>38</td>
</tr>
<tr>
<td>2.7. Replicating Data Out of an Existing Cluster</td>
<td>40</td>
</tr>
<tr>
<td>2.8. Replicating Data Into an Existing Dataservice</td>
<td>42</td>
</tr>
<tr>
<td>2.9. Deploying Additional Datasources, Managers, or Connectors</td>
<td>45</td>
</tr>
<tr>
<td>2.9.1. Adding Datasources to an Existing Deployment</td>
<td>45</td>
</tr>
<tr>
<td>2.9.2. Adding Active Witnesses to an Existing Deployment</td>
<td>46</td>
</tr>
<tr>
<td>2.9.3. Adding Passive Witnesses to an Existing Deployment</td>
<td>46</td>
</tr>
<tr>
<td>2.9.4. Adding Connectors to an Existing Deployment</td>
<td>47</td>
</tr>
<tr>
<td>2.9.5. Adding a remote Composite Cluster</td>
<td>47</td>
</tr>
<tr>
<td>2.10. Removing Datasources, Managers or Connectors</td>
<td>47</td>
</tr>
<tr>
<td>2.10.1. Removing a Datasource from an Existing Deployment</td>
<td>48</td>
</tr>
<tr>
<td>2.10.2. Removing a Connector from an Existing Deployment</td>
<td>49</td>
</tr>
<tr>
<td>2.11. Deploying Tungsten Connector Only</td>
<td>50</td>
</tr>
<tr>
<td>2.12. Starting and Stopping Contingent Tungsten</td>
<td>52</td>
</tr>
<tr>
<td>2.12.1. Restarting the Replicator Service</td>
<td>52</td>
</tr>
<tr>
<td>2.12.2. Restarting the Connector Service</td>
<td>52</td>
</tr>
<tr>
<td>2.12.3. Restarting the Manager Service</td>
<td>53</td>
</tr>
<tr>
<td>2.13. Configuring Startup on Boot</td>
<td>53</td>
</tr>
<tr>
<td>2.14. Upgrading Contingent Tungsten</td>
<td>53</td>
</tr>
<tr>
<td>2.14.1. Upgrading using the Staging Method (with ssh Access)</td>
<td>54</td>
</tr>
<tr>
<td>2.14.2. Upgrading when using INI-based configuration, or without ssh Access</td>
<td>55</td>
</tr>
<tr>
<td>2.14.3. Upgrading from Contingent Tungsten 1.5.3/1.5.4 to Contingent Tungsten 2.0</td>
<td>56</td>
</tr>
<tr>
<td>2.14.4. Installing an Upgraded JAR Patch</td>
<td>57</td>
</tr>
<tr>
<td>2.15. Downgrading to an Earlier Release</td>
<td>57</td>
</tr>
<tr>
<td>3. Advanced Deployments</td>
<td>60</td>
</tr>
</tbody>
</table>
7.6. The labelCommand .......................................................... 164
7.7. The tcm Command .......................................................... 164
7.7.1. tcm list Command ....................................................... 165
7.7.2. tcm index Command .................................................... 167
7.7.3. tcm purge Command ................................................... 168
7.7.4. tcm info Command ...................................................... 168
7.7.5. tcm help Command ..................................................... 168
7.8. The trepctl Command ...................................................... 168
7.8.1. trepctl Options ......................................................... 169
7.8.2. trepctl Global Commands ............................................ 170
7.8.2.1. trepctl kill Command ............................................. 170
7.8.2.2. trepctl services Command ....................................... 171
7.8.2.3. trepctl shutdown Command ..................................... 172
7.8.2.4. trepctl version Command ........................................ 172
7.8.3. trepctl Service Commands ......................................... 173
7.8.3.1. trepctl backup Command ......................................... 173
7.8.3.2. trepctl capabilities Command ................................... 174
7.8.3.3. trepctl check Command .......................................... 175
7.8.3.4. trepctl clear Command .......................................... 175
7.8.3.5. trepctl clients Command ........................................ 175
7.8.3.6. trepctl flush Command .......................................... 176
7.8.3.7. trepctl heartbeat Command ...................................... 176
7.8.3.8. trepctl load Command ............................................ 178
7.8.3.9. trepctl offline Command ......................................... 178
7.8.3.10. trepctl offline-deferred Command ............................. 178
7.8.3.11. trepctl online Command ........................................ 179
7.8.3.12. trepctl properties Command ................................... 182
7.8.3.13. trepctl purge Command ......................................... 182
7.8.3.14. trepctl reset Command ......................................... 183
7.8.3.15. trepctl restore Command ....................................... 183
7.8.3.16. trepctl setrole Command ....................................... 184
7.8.3.17. trepctl shard Command ......................................... 184
7.8.3.18. trepctl start Command .......................................... 185
7.8.3.19. trepctl status Command ........................................ 185
7.8.3.20. trepctl stop Command .......................................... 192
7.8.3.21. trepctl unload Command ....................................... 192
7.8.3.22. trepctl wait Command .......................................... 192
7.9. The tpm Command .......................................................... 193
7.9.1. Comparing Staging and ini tpm Methods ........................... 194
7.9.2. Processing Installs and Upgrades .................................. 196
7.9.3. tpm Staging Configuration .......................................... 197
7.9.3.1. Configuring default options for all services .................. 198
7.9.3.2. Configuring a single service .................................... 198
7.9.3.3. Configuring a single host ....................................... 198
7.9.3.4. Reviewing the current configuration ............................ 198
7.9.3.5. Installation .......................................................... 199
7.9.3.6. Upgrades from a Staging Directory ............................. 199
7.9.3.7. Configuration Changes from a Staging Directory ............ 199
7.9.4. tpm INI File Configuration ......................................... 200
7.9.4.1. Creating an ini file ............................................... 200
7.9.4.2. Installation with INI File ....................................... 201
7.9.4.3. Upgrades with an INI File ...................................... 201
7.9.4.4. Configuration Changes with an INI file ........................ 201
7.9.5. tpm Commands .......................................................... 202
7.9.5.1. tpm configure Command ......................................... 203
7.9.5.2. tpm connector Command ......................................... 203
7.9.5.3. tpm diag Command ................................................ 203
7.9.5.4. tpm fetch Command ............................................... 204
7.9.5.5. tpm firewall Command ............................................ 204
7.9.5.6. tpm help Command ............................................... 204
7.9.5.7. tpm install Command ............................................. 205
7.9.5.8. tpm migrate Command ............................................ 205
7.9.5.9. tpm promote-connector Command ............................... 205
7.9.5.10. tpm query Command ............................................. 205
7.9.5.11. tpm reset Command ............................................. 205
7.9.5.12. tpm reset-thl Command ......................................... 208
7.9.5.13. tpm restart Command ........................................... 208
8. Replication Filters

8.1. Enabling/Disabling Filters
8.2. Enabling Additional Filters
8.3. Filter Status
8.4. Filter Reference
  8.4.1. Bidiremoteslave Filter
  8.4.2. BuildAuditTable Filter
  8.4.3. BuildIndexTable Filter
  8.4.4. CaseMapping Filter
  8.4.5. CDCMetadata Filter
  8.4.6. ColumnName Filter
  8.4.7. ConsistencyCheck Filter
  8.4.8. DatabaseTransform Filter
  8.4.9. Dummy Filter
  8.4.10. EnumToString Filter
  8.4.11. EventMetadata Filter
  8.4.12. Heartbeat Filter
  8.4.13. Logging Filter
  8.4.14. MySQLSessionSupport Filter
  8.4.15. NetworkClient Filter
  8.4.15.1. NetworkClient Configuration
  8.4.15.2. NetworkFilter Protocol
  8.4.15.3. Sample NetworkClient
  8.4.16. OptimizeUpdates Filter
  8.4.17. PrimaryKey Filter
  8.4.18. PrintEvent Filter
  8.4.19. Rename Filter
  8.4.19.1. Rename Filter Examples
  8.4.20. ReplicateColumns Filter
  8.4.21. Replicate Filter
  8.4.22. SetToString Filter
  8.4.23. Shard Filter
  8.4.24. TimeDelay Filter

8.5. JavaScript Filters
  8.5.1. Writing JavaScript Filters
  8.5.1.1. Implementable Functions
  8.5.1.2. Getting Configuration Parameters
  8.5.1.3. Logging Information and Exceptions
  8.5.1.4. Exposed Data Structures
  8.5.2. JavaScript Filter Reference
    8.5.2.1. ansiquotes.js Filter
    8.5.2.2. breadcrumbs.js Filter
    8.5.2.3. dbrename.js Filter
    8.5.2.4. dbselector.js Filter
    8.5.2.5. dbupper.js Filter
    8.5.2.6. dropcolumn.js Filter
    8.5.2.7. dropcomments.js Filter
    8.5.2.8. dropmetatab.js Filter
    8.5.2.9. dropstatemetadata.js Filter
    8.5.2.10. foreignkeychecks.js Filter
    8.5.2.11. insertsonly.js Filter
    8.5.2.12. nocreatedbiinfotexts.js Filter
    8.5.2.13. noonlykeywords.js Filter
    8.5.2.14. pgddl.js Filter
C.4.1. Oracle Environment Variables .................................................................................. 357
C.5. PostgreSQL Database Setup ...................................................................................... 357
D. Terminology Reference ................................................................................................. 358
D.1. Transaction History Log (THL) .................................................................................. 358
D.1.1. THL Format ............................................................................................................ 358
D.2. Generated Field Reference ....................................................................................... 361
D.2.1. Terminology: Fields accessFailures ................................................................. 361
D.2.2. Terminology: Fields active .................................................................................. 361
D.2.3. Terminology: Fields activeSegno ................................................................. 361
D.2.4. Terminology: Fields appliedLastEventId .......................................................... 361
D.2.5. Terminology: Fields appliedLastSegno .............................................................. 361
D.2.6. Terminology: Fields appliedLatency ................................................................. 361
D.2.7. Terminology: Fields appliedClass ................................................................. 361
D.2.8. Terminology: Fields appliedName ................................................................. 361
D.2.9. Terminology: Fields applyTime ....................................................................... 361
D.2.10. Terminology: Fields autoRecoveryEnabled .................................................... 361
D.2.11. Terminology: Fields autoRecoveryTotal .......................................................... 361
D.2.12. Terminology: Fields averageBlockSize ........................................................... 361
D.2.13. Terminology: Fields blockCommitRowCount ................................................... 361
D.2.15. Terminology: Fields channel ........................................................................... 361
D.2.16. Terminology: Fields channels ......................................................................... 361
D.2.17. Terminology: Fields clusterName ................................................................. 361
D.2.18. Terminology: Fields commits ......................................................................... 361
D.2.19. Terminology: Fields committedMinSegno ...................................................... 361
D.2.20. Terminology: Fields criticalPartition ............................................................. 361
D.2.21. Terminology: Fields currentBlockSize ............................................................ 361
D.2.22. Terminology: Fields currentEventId ............................................................... 361
D.2.23. Terminology: Fields currentLastEventId ......................................................... 361
D.2.24. Terminology: Fields currentLastFragno .......................................................... 361
D.2.25. Terminology: Fields currentLastSegno ............................................................ 361
D.2.27. Terminology: Fields dataServerHost ............................................................... 361
D.2.28. Terminology: Fields discardCount ................................................................. 361
D.2.29. Terminology: Fields docChecksum ................................................................. 361
D.2.30. Terminology: Fields estimatedOfflineInterval .............................................. 361
D.2.31. Terminology: Fields eventCount ..................................................................... 361
D.2.32. Terminology: Fields extensions ....................................................................... 361
D.2.33. Terminology: Fields extractTime ..................................................................... 361
D.2.34. Terminology: Fields extractorClass .............................................................. 361
D.2.35. Terminology: Fields extractorName .............................................................. 361
D.2.36. Terminology: Fields filterAClass ................................................................. 361
D.2.37. Terminology: Fields filterAName ................................................................. 361
D.2.38. Terminology: Fields filterTime ...................................................................... 361
D.2.39. Terminology: Fields flushIntervalMillis ...................................................... 361
D.2.40. Terminology: Fields fsyncOnFlush ................................................................. 361
D.2.41. Terminology: Fields headSegno ...................................................................... 361
D.2.42. Terminology: Fields intervalGuard .................................................................. 361
D.2.43. Terminology: Fields lastCommittedBlockSize ................................................ 361
D.2.44. Terminology: Fields lastCommittedBlockTime .............................................. 361
D.2.45. Terminology: Fields latestEpochNumber ...................................................... 361
D.2.46. Terminology: Fields logConnectionTimeout .................................................. 361
D.2.47. Terminology: Fields logDir ............................................................................ 361
D.2.48. Terminology: Fields logFileRetainMillis ....................................................... 361
D.2.49. Terminology: Fields logFileSize ...................................................................... 361
D.2.50. Terminology: Fields masterConnectUri .......................................................... 361
D.2.51. Terminology: Fields masterListenUri .............................................................. 361
D.2.52. Terminology: Fields maxChannel .................................................................... 361
D.2.53. Terminology: Fields maxDelayInterval .......................................................... 361
D.2.54. Terminology: Fields maxOfflineInterval ....................................................... 361
D.2.55. Terminology: Fields maxSize ........................................................................ 361
D.2.56. Terminology: Fields maxSegmentedSeqNo ................................................... 361
D.2.57. Terminology: Fields maximumStoredSegNo .................................................. 361
D.2.58. Terminology: Fields name .............................................................................. 361
D.2.59. Terminology: Fields offlineRequests .............................................................. 361
D.2.60. Terminology: Fields otherTime ........................................................................ 361
D.2.61. Terminology: Fields pendingError ................................................................... 361
List of Figures

2.1. Witness: Active Service ........................................................................................................ 21
2.2. Witness: Inactive Service ...................................................................................................... 22
2.3. Topologies: Master/Slave .................................................................................................... 26
2.4. Topologies: Multisite, Multimaster ...................................................................................... 29
2.5. Topologies: Composite ......................................................................................................... 38
2.6. Topologies: Master/Slave with Replicator .......................................................................... 40
2.7. Topologies: Replicating into a Dataservice ........................................................................ 42
3.1. Migration: Migrating Native Replication using a New Service ........................................ 62
4.1. Sequence: Shunning a Datasource ...................................................................................... 93
5.1. Tungsten Connector Basic Architecture ............................................................................ 127
5.2. Basic MySQL/Application Connectivity ............................................................................. 128
5.3. Advanced MySQL/Application Connectivity ..................................................................... 129
5.4. Using Tungsten Connector for MySQL/Application Connectivity .................................... 129
5.5. Tungsten Connector during a failed datasource ................................................................. 130
5.6. Tungsten Connector routing architecture .......................................................................... 130
5.7. Tungsten Connector Bridge Mode Architecture ............................................................... 137
5.8. Tungsten Connector Authentication ................................................................................... 138
7.1. tpm Staging Based Deployment ......................................................................................... 194
7.2. tpm INI Based Deployment ............................................................................................... 195
8.1. Filters: Pipeline Stages on Masters ................................................................................... 259
8.2. Filters: Pipeline Stages on Slaves ..................................................................................... 260
C.1. Tungsten Deployment ......................................................................................................... 347
List of Tables

1.1. Key Terminology .................................................................................................................. 18
2.1. Key Terminology .................................................................................................................. 19
5.1. Routing Method Selection ..................................................................................................... 131
5.2. Inline Interface Commands .................................................................................................. 147
6.1. Manager REST API Endpoints ............................................................................................. 152
7.1. check_tungsten_services Options ...................................................................................... 153
7.2. check_tungsten_online Options .......................................................................................... 154
7.3. check_tungsten_latency Options ....................................................................................... 155
7.4. check_tungsten_progress Options ..................................................................................... 155
7.5. cctrl Command-line Options ............................................................................................. 156
7.6. cctrl Commands ................................................................................................................. 157
7.7. cctrldatatable Commands ................................................................................................. 160
7.8. cluster_backup Options ...................................................................................................... 164
7.9. thl Options ......................................................................................................................... 165
7.10. trepctl Command-line Options ......................................................................................... 169
7.11. trepctl Replicator Wide Commands .................................................................................. 170
7.12. trepctl Service Commands ............................................................................................... 173
7.13. trepctl backup Command Options .................................................................................... 173
7.14. trepctl clients Command Options .................................................................................... 175
7.15. trepctl offline-deferred Command Options .................................................................... 178
7.16. trepctl online Command Options .................................................................................... 179
7.17. trepctl purge Command Options ...................................................................................... 183
7.18. trepctl reset Command Options ....................................................................................... 183
7.19. trepctl setrole Command Options ................................................................................... 184
7.20. trepctl shard Command Options ...................................................................................... 184
7.21. trepctl status Command Options ..................................................................................... 185
7.22. trepctl wait Command Options ......................................................................................... 192
7.23. TPM Deployment Methods ............................................................................................... 195
7.24. tpm Common Options ....................................................................................................... 197
7.25. tpm Core Options .............................................................................................................. 202
7.26. tpm Commands ................................................................................................................ 202
7.27. tpm Configuration Options ............................................................................................... 211
7.28. tungsten_provision_slave Command-line Options ......................................................... 252
7.29. tungsten_monitor Command-line Options ....................................................................... 253
7.30. tungsten_read_master_events Command-line Options .................................................. 255
7.31. tungsten_set_position Command-line Options ............................................................... 256
7.32. tungsten_health_check Command-line Options ............................................................... 257
D.1. THL Event Format ............................................................................................................... 359
E.1. Continuent Tungsten Directory Structure .......................................................................... 369
E.2. Continuent Tungsten Sub-Directory Structure .................................................................... 376
Preface

This manual documents Continuent Tungsten 2.0 up to and including 2.0.5 build 3. Differences between minor versions are highlighted stating the explicit minor release version, such as 2.0.5.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 --hosts` links automatically to the corresponding reference page.

- Parameters or information explicitly used to set values to commands or options is formatted using **this style**.

- Option values, for example on the command-line are marked up using **this 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 **Text in this style**. Commands are also automatically linked to their respective documentation page when this is known. For example, `tpm` links automatically to the corresponding reference page.

- **Text in this style** indicates literal or character sequence text used to show a specific value.

- Filenames, directories or paths are shown like this `/etc/passwd`. Filenames and paths are automatically linked to the corresponding reference page if available.

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

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

```
shell> cd /opt/staging
shell> unzip continuum-tungsten-2.0.5-3.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.
If a command-line or program listing entry contains lines that are two wide to be displayed within the documentation, they are marked using the » character:

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

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

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

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

3. Quickstart Guide

- Are you planning on completing your first installation?
  - Do you know the Section 2.2, "Requirements"?
  - Have you followed the Appendix C, Prerequisites?
  - Have you decided which installation method you will use? INI or Staging?
  - Have you chosen which deployment type from Chapter 2, Deployment? Is this a Master/Slave deployment?

- 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 7.9.1, "Comparing Staging and INI tpm Methods".

- Do you want to upgrade to the latest version?
  - See Section 7.9.5.17, "tpm update Command".

- Are you trying to update or change the configuration of your system?
  - See Section 7.9.5.17, "tpm update Command".

- Has your system suffered a failure?
  - For recovery methods and instructions, see Section 4.5, "Datasource Recovery Steps".

- Would you like to perform database or operating system maintenance?
  - See Section 4.11, "Performing Database or OS Maintenance".

- Do you need to backup or restore your system?
  - For backup instructions, see Section 4.9, "Creating a Backup", and to restore a previously made backup, see Section 4.10, "Restoring a Backup".
Chapter 1. Introduction

Continuent Tungsten™ provides a suite of tools to aid the deployment of database clusters using MySQL. Continuent Tungsten™ consists of three primary tools:

- **Tungsten Replicator**

  Tungsten Replicator supports replication between different databases. Tungsten Replicator acts as a direct replacement for the native MySQL replication, in addition to supporting connectivity to PostgreSQL, Oracle, MongoDB, Vertica and others.

- **Tungsten Connector**

- **Tungsten Manager**

1.1. Tungsten Replicator

Tungsten Replicator is an open source 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, PostgreSQL, MongoDB and Vertica, among others.

Understanding the Tungsten Replicator works requires looking at the overall replicator structure. In the diagram below is the top-level overview of the structure of a replication service.

At this level, there are three major components in the system that provide the core of the replication functionality:

- **Extractor**

  The extractor component reads data from a data server, such as MySQL or Oracle, 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.

  For example, within MySQL, information is read directly from the binary log that MySQL produces for native replication; in Oracle, the Change Data Capture (CDC) information is used as the information source.

- **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 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. Transaction History Log (THL)

Tungsten Replicator operates by reading information from the source database (MySQL, PostgreSQL, Oracle) and transferring that information to the Tungsten 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 option 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 an 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.

1.2. Tungsten Manager

The Tungsten Manager is responsible for monitoring and managing a Continuent Tungsten dataservice. The manager has a number of control and supervisory roles for the operation of the cluster, and acts both as a control and a central information source for the status and health of the dataservice as a whole.

Primarily, the Tungsten Manager handles the following tasks:

- Monitors the replication status of each datasource within the cluster.
- Communicates and updates Tungsten Connector with information about the status of each datasource. In the event of a change of status, Tungsten Connectors are notified so that queries can be redirected accordingly.
- Manages all the individual components of the system. Using the Java JMX system the manager is able to directly control the different components to change status, control the replication process, and
- Includes an advanced rules engine. The rule engine is used to respond to different events within the cluster and perform the necessary operations to keep the dataservice in optimal working state. During any change in status, whether user-selected or automatically triggered due to a failure, the rules are used to make decisions about whether to restart services, swap masters, or reconfigure connectors.

1.3. Tungsten Connector

The Tungsten Connector is a service that sits between your application server and your MySQL or PostgreSQL database. The connector routes connections from your application servers to the data sources within the cluster, automatically distributing and redirecting queries to each datasource according to load balancing and availability requirements.

The primary goal of Tungsten Connector is to effectively route and redirect queries between the master and slave data sources within the cluster. Client applications talk to the connector, while the connector determines where the packets should really go, depending on the scaling and availability. Using a connector in this way effectively hides the complexities of the cluster size and configuration, allowing your cluster to grow and shrink without interrupting your client application connectivity. Client applications remain connected even though the number, configuration and orientation of the slaves within the cluster may change.

During failover or system maintenance Tungsten Connector takes information from Tungsten Manager to determine which hosts are up and available, and redirects queries only to those servers that are online within the cluster.

For load balancing, Tungsten Connector supports a number of different solutions for redirecting queries to the different data sources within the network. Solutions are either based on explicit routing, or an implied or automatic read/write splitting mode where data is automatically distributed between master hosts (writes) and slave hosts (reads).

Basic read/write splitting uses packet inspection to determine whether a query is a read operation (SELECT) or a write (INSERT, UPDATE, DELETE). The actual selection mechanism can be fine-tuned using the different modes according to your application requirements.

The supported modes are:

- SmartScale
With SmartScale, data is automatically distributed among the datasources using read/write splitting. Where possible, the connector selects read queries by determining how up to date the slave is, and using a specific session model to determine which host is up to date according to the session and replication status information. Session identification can be through predefined session types or user-defined session strings.

- **Direct Reads**
  Direct reads uses the read/write splitting model, but directs read queries to dedicated read-only connections on the slave. No attempt is made to determine which host may have the most up to date version of the data. Connections are pooled between the connector and datasources, and this results in very fast execution.

- **Host Based Routing**
  Explicit host based routing uses different IP addresses on datasources to identify whether the operation should be directed to a master or a slave. Each connector is configured with two IP addresses, connecting to one IP address triggers the connection to be routed to the current master, while connecting to the second IP routes queries to a slave.

- **SQL Based Routing**
  SQL based routings employs packet inspection to identify key strings within the query to determine where the packets should be routed.

These core read/write splitting modes can also be explicitly overridden at a user or host level to allow your application maximum flexibility.

Internally, Tungsten Connector supports the native protocol (for both MySQL and PostgreSQL), and accepts the raw packet data from the client and sends those packets directly to the datasource. Because it is the native network packets that are being forwarded between hosts the performance is kept high, without requiring any additional overhead or intelligence within the application.

The connector handles the distribution of packets between datasources, allowing clients to remain connected to Tungsten Connector even while the underlying datasources may become disconnected, or expanded as new datasources are added to the cluster.

### 1.4. Key Terminology

Continuent Tungsten uses key terminology for different components in the system. These are used to distinguish specific elements of the overall system at the different levels of operations.

**Table 1.1. Key Terminology**

<table>
<thead>
<tr>
<th>Continuent Term</th>
<th>Traditional Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>datasource</td>
<td>Cluster</td>
<td>The collection of machines that make up a single Tungsten Dataservice. Individual hosts within the dataservice are called datasources. Each dataservice is identified by a unique name, and multiple dataservices can be managed from one server.</td>
</tr>
<tr>
<td>dataserver</td>
<td>Database</td>
<td>The database on a host. Datasources include MySQL, PostgreSQL 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>connector</td>
<td></td>
<td>A connector is a connection to a dataservice and provides connectivity to the underlying database for clients.</td>
</tr>
<tr>
<td>staging host</td>
<td></td>
<td>The machine (and directory) from which Continuent Tungsten™ is installed and configured. The machine does not need to be the same as any of the existing hosts in the dataservice.</td>
</tr>
<tr>
<td>active witness</td>
<td></td>
<td>A machine in the dataservice that runs the manager process but is not running a database server. This server will be used to establish quorum in the event that a datasource becomes unavailable.</td>
</tr>
<tr>
<td>coordinator</td>
<td></td>
<td>The datasource or active witness in a dataservice that is responsible for making decisions on the state of the dataservice. The coordinator is usually the member that has been running the longest. It will not always be the master. When the manager process on the coordinator is stopped, or no longer available, a new coordinator will be chosen from the remaining members.</td>
</tr>
</tbody>
</table>

### 1.5. Datasource Types
Chapter 2. Deployment

Creating a Continuent Tungsten Dataservice using Continuent Tungsten combines a number of different components, systems, and functionality, to support a running database dataservice that is capable of handling database failures, complex replication topologies, and management of the client/database connection for both load balancing and failover scenarios.

How you choose to deploy depends on your requirements and environment. All deployments operate through the `tpm` command. `tpm` operates in two different modes:

- **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 Continuent Tungsten 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` to configure the service on the local host. The `ini` file must be create on each host that will be part of the cluster. `tpm` only manages the services on the local host; in a multi-host deployment, upgrades, updates, and configuration must be handled separately on each host.

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

### 2.1. Host Types

Before covering the basics of creating different dataservice types, 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>composite dataservice</td>
<td>Multi-Site Cluster</td>
<td>A configured Continuent Tungsten service consisting of multiple dataservices, typically at different physical locations.</td>
</tr>
<tr>
<td>dataservice</td>
<td>Cluster</td>
<td>A configured Continuent Tungsten service consisting of dataservers, datasources and connectors.</td>
</tr>
<tr>
<td>datasource</td>
<td>Database</td>
<td>The database on a host. Datasources include MySQL, PostgreSQL or Oracle.</td>
</tr>
<tr>
<td>staging host</td>
<td>-</td>
<td>The machine from which Continuent Tungsten 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>
<tr>
<td>connector</td>
<td>-</td>
<td>A connector is a routing service that provides management for connectivity between application services and the underlying dataserver.</td>
</tr>
<tr>
<td>Witness host</td>
<td>-</td>
<td>A witness host is a host that can be contacted using the <code>ping</code> protocol to act as a network check for the other nodes of the cluster. Witness hosts should be on the same network and segment as the other nodes in the dataservice.</td>
</tr>
</tbody>
</table>

#### 2.1.1. Manager Hosts

The manager plays a key role within any dataservice, communicating between the replicator, connector and datasources to understand the current status, and controlling these components to handle failures, maintenance, and service availability.

The primary role of the manager is to monitor each of the services, identify problems, and react to those problems in the most effective way to keep the dataservice active. For example, in the case of a datasource failure, the datasource is temporarily removed from the cluster, the connector is updated to route queries to another available datasource, and the replication is disabled.

These decisions are driven by a rule-based system, which checks current status values, and performs different operations to achieve the correct result and return the dataservice to operational status.

In terms of control and management, the manager is capable of performing backup and restore information, automatically recovering from failure (including re-provisioning from backups), and is also able to individually control the configuration, service startup and shutdown, and overall control of the system.

Within a typical Continuent Tungsten deployment there are multiple managers and these keep in constant contact with each other, and the other services. When a failure occurs, multiple managers are involved in decisions. For example, if a host is no longer visible to one
Deployment

manager, it does not make the decision to disable the service on its own; only when a majority of managers identify the same result is
the decision made. For this reason, there should be an odd number of managers (to prevent deadlock), or managers can be augmented
through the use of witness hosts.

One manager is automatically installed for each configured datasource; that is, in a three-node system with a master and two slaves, three
managers will be installed.

Checks to determine the availability of hosts are performed by using either the Echo TCP/IP protocol on port 7 (default), or using the
system ping protocol to determine whether a host is available. The configuration of the protocol to be used can be made by adjusting the
manager properties. For more information, see Section C.2.2.3, “Host Availability Checks”.

2.1.2. Connector (Router) Hosts

Connectors (known as routers within the dataservice) provide a routing mechanism between client applications and the dataservice. The
Tungsten Connector component automatically routes database operations to the master or slave, and takes account of the current cluster
status as communicated to it by the Tungsten Manager. This functionality solves three primary issues that might normally need to be
handled by the client application layer:

• Datasource role redirection (i.e. master and slave). This includes read/write splitting, and the ability to read data from a slave that is up to
date with a corresponding write.

• Datasource failure (high-availability), including the ability to redirect client requests in the event of a failure or failover. This includes
maintenance operations.

• Dataservice topology changes, for example when expanding the number of datasources within a dataservice

The primary role of the connector is to act as the connection point for applications that can remain open and active, while simultaneously
supporting connectivity to the datasources. This allows for changes to the topology and active role of individual datasources without
interrupting the client application. Because the operation is through one or more static connectors, the application also does not need to
be modified or changed when the number of datasources is expanded or altered.

Depending on the deployment environment and client application requirements, the connector can be installed either on the client
application servers, the database servers, or independent hosts. For more information, see Section 5.7, “Clients and Deployment”.

Connectors can also be installed independently on specific hosts. The list of enabled connectors is defined by the --connectors option to tpm. A Continuent Tungsten dataservice can be installed with more connector servers than datasources or managers.

2.1.3. Replicator Hosts

The replicator provides the core replication of information between datasources and, in composite deployment, between dataservices.
The replicator operates by extracting data from the ‘master’ datasource (for example, using the MySQL binary log), and then applies the
data to one or more target datasources.

Different deployments use different replicators and configurations, but in a typical Continuent Tungsten deployment a master/slave or
multimaster deployment model is used. For Continuent Tungsten deployments there will be one replicator instance installed on each
datasource host.

Within the dataservice, the manager controls each replicator service and it able to alter the replicator operation and role, for example by
switching between master and slave roles. The replicator also provides information to the manager about the latency of the replication
operation, and uses this with the connectors to control client connectivity into the dataservice.

Replication within Continuent Tungsten is supported by Tungsten Replicator™ and this supports a wide range of additional deployment
topologies, and heterogeneous deployments including MongoDB, Vertica, and Oracle. Replication to and from a dataservice are
supported. For more information on replicating out of an existing dataservice, see Section 2.7, “Replicating Data Out of an Existing
Cluster” or ???.

Replicators are automatically configured according to the datasources and topology specified when the dataservice is created.

2.1.4. Witness Hosts

Continuent Tungsten operates through the rules built into the manager that make decisions about different configuration and status
settings for all the services within the cluster. In the event of a communication failure within the system it is vital for the manager, in
automatic policy mode, to perform a switch from a failed or unavailable master.

Within the network, the managers communicate with each other, in addition to the connectors and dataservers to determine their
availability. The managers compare states and network connectivity. In the event of an issue, managers ‘vote’ on whether a failover or
switch should occur.

The rules are designed to prevent unnecessary switches and failovers. Managers vote, and an odd number of managers helps to ensure
that prevent split-brain scenarios when invalid failover decisions have been made.
Two types of witness are supported:

- **Passive Witness** — a passive witness is checked by the managers using a network ping to determine if the host is available. The witness host or hosts are used only as check to verify whether a failed host or failed network may be to blame. Because

- **Active Witness** — an active witness is an instance of Tungsten Manager running on a host that is otherwise not part of the dataservice. An active witness has full voting rights within the managers and can therefore make informed decisions about the dataservice state in the event of a failure. Active witnesses can only be a member of one cluster at a time.

All managers are active witnesses, and active witnesses are the recommended solution for deployments where network availability is less certain (i.e. cloud environments), and where you have two-node deployments.

**Continuent Tungsten Quorum Requirements**

- There should be at least three managers (including any active witnesses)
- There should be an odd number of managers and witnesses, to prevent deadlocks.
- If the dataservice contains only two hosts, at least one active witness must be installed.
- Dataservices may contain either passive or active witnesses, but not both.

These rules apply for all Continuent Tungsten installations and must be adhered to. Deployment will fail if these conditions are not met.

The rules for witness selection are as follows:

1. Passive witnesses must be on the same network segment the managers. To prevent issues where a network switch or router failure would cause the managers to falsely identify a network failure, the managers must be able to connect to each other without having to route across networks or network segments.

Active witnesses can be located beyond or across network segments, but all active witnesses must have clear communication channel to each other, and other managers. Difficulties in contacting other managers and services in the network could cause unwanted failover or shunning of datasources.

For example, consider the following scenario:

- Master dataserver on **hostA**, with slave dataservers on **hostB** and **hostC**
- Manager on **hostA** can see the dataserver on **hostA** and **hostB**, but not **hostC**
- Manager on **hostB** can see the dataserver on **hostB** and **hostC**, but not **hostA**
- Manager on **hostC** can see the dataserver on **hostA**, **hostB** and **hostC**
- Manager on **hostA**, **hostB**, and **hostC** can communicate with each other

Figure 2.1. Witness: Active Service

The master will not be automatically switched, given that **hostA** is still available to two of the managers in the network.

If a second manager identifies **hostA** has failed:
Passive witnesses can be enabled when using `tpm` by using the `--witnesses` option:

```
shell> /tools/tpm install alpha --witnesses=hostC,hostD
```

To enable active witnesses, the `--enable-active-witnesses=true` option must be specified and the hosts that will act as active witnesses must be added to the list of hosts provided to `--members`. This enables all specified witnesses to be enabled as active witnesses:

```
shell> /tools/tpm install alpha --enable-active-witnesses=true
    --witnesses=hostC
    --members=hostA,hostB,hostC
```

### 2.2. Requirements

#### 2.2.1. 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.</td>
</tr>
<tr>
<td>Linux</td>
<td>Ubuntu</td>
<td>Primary platform</td>
<td>Ubuntu 9.x/10.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>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 Leopard and Snow Leopard are used for development at Continuent but not certified. We will fix any bugs reported by customers.</td>
</tr>
<tr>
<td>Windows</td>
<td>Limited Support</td>
<td></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>Limited Support</td>
<td></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>

#### 2.2.2. 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</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>
</tbody>
</table>
### Deployment

<table>
<thead>
<tr>
<th>Database</th>
<th>Version</th>
<th>Support Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Percona</td>
<td>5.5, 5.6</td>
<td>Primary platform</td>
<td></td>
</tr>
<tr>
<td>MariaDB</td>
<td>5.5</td>
<td>Primary platform</td>
<td></td>
</tr>
<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>
<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>

#### 2.2.3. 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.

- **Tungsten Manager** requires approximately 500MB of VM space for execution.

#### 2.2.4. 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 an INI-file based deployment, this space will be required on each server. For more information on the different methods, see Section 7.9.1, "Comparing Staging and INI tp 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 E.1.4, "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 Continuent Tungsten is recommended to ensure that a full disk does not impact your OS or DBMS. Because the replicator reads and writes information using buffered I/O in a serial fashion, spinning disk and Network Attached Storage (NAS) is suitable for storing THL, as there is no random-access or seeking.

#### 2.2.5. Java Requirements

Tungsten Replicator is known to work with Java 1.6. and Java 1.7 and using the following JVMs:

- Oracle JVM/JDK 6
- Oracle JVM/JDK 7
- OpenJDK 6
- OpenJDK 7

#### 2.2.6. Cloud Deployment Requirements

Cloud deployments require a different set of considerations over and above the general requirements. The following is a guide only, and where specific cloud environment requirements are known, they are explicitly included:

**Instance Types/Configuration**
### Attribute Guidance

<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>\texttt{m1.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.
- When using Active Witnesses, a \texttt{micro} instance can be used for a single cluster. For composite clusters, an instance size larger than \texttt{micro} must be used.
- Multiple EBS-optimized volumes for data, using Provisioned IOPS for the EBS volumes depending on workload:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>\texttt{tpm} Option/Value</th>
<th>MySQL \texttt{my.cnf} Option/Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL Data</td>
<td>datasource-mysql-data-directories [227] /volumes/mysql/data datadir /volumes/mysql/data</td>
<td></td>
</tr>
<tr>
<td>MySQL Binary Logs</td>
<td>datasource-log-directories [227] /volumes/mysql/binlogs log-bin /volumes/mysql/binlogs/mysql-bin</td>
<td></td>
</tr>
<tr>
<td>Transaction History Logs (THL)</td>
<td>thl-directories [250] /volumes/mysql/thl</td>
<td></td>
</tr>
</tbody>
</table>

### Recommended Replication Formats

- \texttt{MIXED} is recommended for MySQL master/slave topologies (e.g., either single clusters or primary/data-recovery setups).
- \texttt{ROW} is strongly recommended for multi-master setups. Without \texttt{ROW}, data drift is a possible problem when using \texttt{MIXED} or \texttt{STATEMENT}. Even with \texttt{ROW} there are still cases where drift is possible but the window is far smaller.
- \texttt{ROW} is required for heterogeneous replication.

### 2.3. Deployment Sources

Continuent Tungsten is available in a number of different distribution types, and the methods for configuration available for these different packages differs. See Section 7.9.1, “Comparing Staging and TAR \texttt{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>\texttt{tpm} Command-line Configuration</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>\texttt{tpm} INI File Configuration</td>
<td>Yes</td>
<td></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:
Deployment

The version number is 2.0.5 and build number 3. Build numbers indicate which build a particular release version is based on, and may be useful when installing patches provided by support.

2.3.1. Using the TAR/GZipped files

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

```
shell> cd /opt/continuent/software
shell> tar zxf continuent-tungsten-2.0.5-3.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 continuent-tungsten-2.0.5-3
```

2.3.2. 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 exist

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 C, Prerequisites.

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

**Note**

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

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

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

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

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

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

### 2.4. Deploying a Master/Slave Topology

The creation of a master/slave dataservice is achieved by using the `tpm` command. `tpm` performs all of the operations required to get your dataservice, connectors, and replication service configured. In the mode below, the command will create and start the service.

**Figure 2.3. Topologies: Master/Slave**

Within a master/slave service, there is a single master which replicates data to the slaves. The Tungsten Connector handles connectivity by the application and distributes the load to the dataservers in the dataservice.

#### 2.4.1. Installing a Master/Slave Configuration

To perform a basic installation in a master/slave topology:

1. Download the Continuent Tungsten tarball, and unpack it:
   ```
   shell> tar zxf continuent-tungsten-2.0.5-3.tar.gz
   ```

2. Change to the Continuent Tungsten directory:
   ```
   shell> cd continuent-tungsten-2.0.5-3
   ```

3. Run `tpm` to perform the installation. This method assumes you are using the Section 7.9.3, "tpm Staging Configuration" method:
   ```
   shell> ./tools/tpm install alpha \
   --user=tungsten \
   --install-directory=/opt/continuent \
   --members=host1,host2,host3 \
   --connectors=host1,host2,host3 \
   --master=host1 \
   --datasource-user=tungsten \
   --datasource-password=password \
   --application-user=app_user \
   ```
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
  This runs the tpm command. install [234] indicates that we are installing and creating a new dataservice, and alpha is the name and identity of the dataservice being created.
  - --user=tungsten [251]
    The operating system user name that you have created for the Tungsten service, tungsten.
  - --install-directory=/opt/continuent [234]
    The installation directory of the Tungsten service. This is where the service will be installed on each server in your dataservice.
  - --members=host1,host2,host3 [237]
    A comma separated list of all the hosts that are part of this dataservice.
  - --connectors=host1,host2,host3 [226]
    A comma separated list of the hosts that will have a connector service created on them.
  - --master=host1 [237]
    The hostname of the server that will be the master MySQL server.
  - --dataservice-witnesses=witness [252]
    The hostname of a computer that will be contacted using ping in the event of a network problem.
  - --datasource-user=tungsten [246]
    The MySQL user name to use when connecting to the MySQL database.
  - --datasource-password=password [245]
    The MySQL password for the user that will connect to the MySQL database.
  - --application-user=app_user [221]
    The application user name.
  - --application-password=password [220]
    The application password.
  - --datasource-port=13306 [246]
    The TCP/IP port that the MySQL database is listening on for connections.
  - --connector-listen-port=3306 [220]
    The TCP/IP port on which to listen for incoming connections by the Tungsten connector service. To emulate the standard MySQL database service, port 3306 is used.
  - --start-and-report [248]
    Tells tpm to startup the service, and report the current configuration and status.

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 the service starts correctly, you should see the configuration and current status of the service.

### 2.4.2. Managing and Monitoring a Master/Slave Configuration

Once the creation of your dataservice completes successfully, the output will show the dataservice status:
The information will be output for each of the nodes within the dataservice to confirm that the structure and information is identical on each host.

Once your dataservice is configured and up and running, you can now configure your clients to connect to the configured connector hosts. Data inserted and updated into the database through the connector will be automatically sent to the master within the dataservice, and reads will be distributed to the slaves.

### 2.5. Deploying a Multisite/Multimaster Topology

A Multisite/Multimaster topology provides all the benefits of a typical dataservice at a single location, but with the benefit of also replicating the information to another site. The underlying configuration within Continuent Tungsten uses the Tungsten Replicator System of Record (SOR) service, which enables multimaster operation between the two sites.

The configuration is in two separate parts:

- Continuent Tungsten dataservice that operates the main dataservice service within each site.
- Tungsten Replicator dataservice that provides replication between the two sites; one to replicate from site1 to site2, and one for site2 to site1.

A sample display of how this operates is provided in Figure 2.4, "Topologies: Multisite, Multimaster".
The service can be described as follows:

- **Continuent Tungsten Service: east**
  Replicates data between east1, east2 and east3 (not shown).

- **Continuent Tungsten Service: west**
  Replicates data between west1, west2 and west3 (not shown).

- **Tungsten Replicator Service: east**
  Defines the replication of data within east as a replicator service using Tungsten Replicator. This service reads from all the hosts within the Continuent Tungsten service `east` and writes to `west1`, `west2`, and `west3`. The service name is the same to ensure that we do not duplicate writes from the clustered service already running.

  Data is read from the `east` Continuent Tungsten and replicated to the `west` Continuent Tungsten dataservice. The configuration allows for changes in the Continuent Tungsten dataservice (such as a switch or failover) without upsetting the site-to-site replication.

- **Tungsten Replicator Service: west**
  Defines the replication of data within west as a replicator service using Tungsten Replicator. This service reads from all the hosts within the Continuent Tungsten service `west` and writes to `east1`, `east2`, and `east3`. The service name is the same to ensure that we do not duplicate writes from the clustered service already running.

  Data is read from the `west` Continuent Tungsten and replicated to the `east` Continuent Tungsten dataservice. The configuration allows for changes in the Continuent Tungsten dataservice (such as a switch or failover) without upsetting the site-to-site replication.

- **Tungsten Replicator Service: east_west**
Replicates data from East to West, using Tungsten Replicator. This is a service alias that defines the reading from the dataservice (as a slave) to other servers within the destination cluster.

- **Tungsten Replicator Service: west_east**

Replicates data from West to East, using Tungsten Replicator. This is a service alias that defines the reading from the dataservice (as a slave) to other servers within the destination cluster.

**Requirements.** Recommended releases are Continuent Tungsten 2.0.1 and Tungsten Replicator 2.2.0.

### 2.5.1. Preparing Hosts for Multimaster

Some considerations must be taken into account for any multimaster scenario:

- For tables that use auto-increment, collisions are possible if two hosts select the same auto-increment number. You can reduce the effects by configuring each MySQL host with a different auto-increment settings, changing the offset and the increment values. For example, adding the following lines to your `my.cnf` file:

  ```
  auto-increment-offset = 1
  auto-increment-increment = 4
  ```

  In this way, the increments can be staggered on each machine and collisions are unlikely to occur.

- Use row-based replication. Statement-based replication will work in many instances, but if you are using inline calculations within your statements, for example, extending strings, or calculating new values based on existing column data, statement-based replication may lead to significant data drift from the original values as the calculation is computed individually on each master. Update your configuration file to explicitly use row-based replication by adding the following to your `my.cnf` file:

  ```
  binlog-format = row
  ```

- Beware of triggers. Triggers can cause problems during replication because if they are applied on the slave as well as the master you can get data corruption and invalid data. Continuent Tungsten cannot prevent triggers from executing on a slave, and in a multimaster topology there is no sensible way to disable triggers. Instead, check at the trigger level whether you are executing on a master or slave. For more information, see Section A.3.1, “Triggers”.

- Ensure that the `server-id` for each MySQL configuration has been modified and is different on each host. This will help to prevent the application of data originating on the a server being re-applied if the transaction is replicated again from another master after the initial replication. Continuent Tungsten is designed not to replicate these statements, and uses the server ID as part of the identification process.

### 2.5.2. Installing Multisite, Multimaster Topologies

Creating the configuration requires two distinct steps, the first to create the two Continuent Tungsten deployments, and a second that creates the Tungsten Replicator configurations on different network ports, and different install directories.

In short, the replication is composed of two groups, one for the Continuent Tungsten and one for Tungsten Replicator. Each involves multiple steps:

1. Configuring the Continuent Tungsten deployment:
   a. Download the Continuent Tungsten tarball, and unpack it:

      ```
      shell> tar zxf continuent-tungsten-2.0.5-3.tar.gz
      ```

   b. Change to the Continuent Tungsten directory:

      ```
      shell> cd continuent-tungsten-2.0.5-3
      ```

   c. **Important**

      The following commands *must* be run from the Continuent Tungsten directory.

      Create the default configuration that will be used for the two individual Continuent Tungsten services:

      ```
      shell> ./tools/tpm configure defaults --reset \
      --user=tungsten \ 
      --install-directory=/opt/continuent \ 
      --replication-user=tungsten \ 
      --replication-password='password' \ 
      ```
d. Create the Continuent Tungsten service for east. This creates a service using the three hosts, \texttt{east1}, \texttt{east2}, and \texttt{east3}, a Tungsten Connector on each node, and with \texttt{east1} as the master:

```
shell> ./tools/tpm configure east \
  --topology=clustered \
  --members=east1,east2,east3 \
  --connectors=east1,east2,east3 \
  --master=east1
```

e. Create the Continuent Tungsten service for west. This creates a service using the three hosts, \texttt{west1}, \texttt{west2}, and \texttt{west3}, a Tungsten Connector on each node, and with \texttt{west1} as the master:

```
shell> ./tools/tpm configure west \
  --topology=clustered \
  --members=west1,west2,west3 \
  --connectors=west1,west2,west3 \
  --master=west1
```

f. Install the services on all six hosts, creating two distinct services. There is no relationship or replication between these two services at this point:

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

The installation will output the status of the two clusters, with one report for each host within each cluster. For example, the output from \texttt{east1} is shown below:

```
Getting dataservice status on east1
Continuent Tungsten 2.0.5 build 3
east: session established
[LOGICAL] /east > ls
COORDINATOR[east1:AUTOMATIC:ONLINE]

ROUTERS:
+----------------------------------------------------------------------------+
|connector@east1[17951](ONLINE, created=0, active=0)                         |
|connector@east2[17939](ONLINE, created=0, active=0)                         |
|connector@east3[17961](ONLINE, created=0, active=0)                         |
+----------------------------------------------------------------------------+

DATASOURCES:
+----------------------------------------------------------------------------+
|east1(master:ONLINE, progress=0, THL latency=1.443)                         |
|STATUS [OK] [2013/11/25 11:24:35 AM GMT]                                    |
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=master, state=ONLINE)                                     |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+

+----------------------------------------------------------------------------+
|east2(slave:ONLINE, progress=0, latency=15.819)                             |
|STATUS [OK] [2013/11/25 11:24:39 AM GMT]                                    |
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=slave, master=east1, state=ONLINE)                        |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+

+----------------------------------------------------------------------------+
|east3(slave:ONLINE, progress=0, latency=13.346)                             |
|STATUS [OK] [2013/11/25 11:24:38 AM GMT]                                    |
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=slave, master=east1, state=ONLINE)                        |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+
```

2. Once the Continuent Tungsten services have been created, the Tungsten Replicator services should be deployed:

a. Download the Tungsten Replicator tarball, and unpack it:

```
shell> tar zxf tungsten-replicator-2.2.1.tar.gz
```
b. Change to the Tungsten Replicator directory:

```
shell> cd tungsten-replicator-2.2.1
```

c. **Important**

The following commands must be run from the Tungsten Replicator directory.

First set the defaults for the Tungsten Replicator services. These values will be shared by all the services. Because the Tungsten Replicator is being installed on the same hosts, it must be configured in a different directory (/opt/replicator), and use a different RMI and THL port:

```
shell> ./tools/tpm configure defaults --reset \\
   --user=tungsten \\
   --install-directory=/opt/replicator \\
   --rmi-port=10002 \\
   --thl-port=2114 \\
   --replication-user=tungsten \\
   --replication-password='password' \\
   --application-user=app_user \\
   --application-password=password \\
   --skip-validation-check=ManagerWitnessNeededCheck \\
   --start-and-report
```

d. Create a Tungsten Replicator service, `east`. This defines the nodes within a clustered service - i.e. an existing dataservice using Continuent Tungsten. This will enable the system to create a connection for the configured dataservice with the configured hosts and use the dataservice as a source or destination for replication.

```
shell> ./tools/tpm configure east \\
   --topology=clustered \\
   --members=east1,east2,east3 \\
   --connectors=east1,east2,east3 \\
   --master=east1
```

The service creates a unique reference that includes all three hosts in the cluster, and all three hosts will be used as the source of data for the replication from dataservice to cluster.

e. Create a Tungsten Replicator service, `west`, that defines the source dataservice of the `west` service:

```
shell> ./tools/tpm configure west \\
   --topology=clustered \\
   --members=west1,west2,west3 \\
   --connectors=west1,west2,west3 \\
   --master=west1
```

f. Create a service, `east_west`, that replicates information from the `east` Continuent Tungsten to the `west` Continuent Tungsten (using the Tungsten Replicator service names `east` and `west`).

The effect of this definition is that the `east` Tungsten Replicator service will read data from the `east` service `(east{1,2,3})` and write to the hosts in the `west` service `(west{1,2,3})`.

```
shell> ./tools/tpm configure east_west \\
   --topology=cluster-slave \\
   --master-dataservice=east \\
   --slave-dataservice=west
```

g. Create a service, `west_east`, that replicates information from the `west` Continuent Tungsten to the `east` Continuent Tungsten.

The effect of this definition is that the `west` Tungsten Replicator service will read data from the `west` service `(west{1,2,3})` and write to the hosts in the `east` service `(east{1,2,3})`.

```
shell> ./tools/tpm configure west_east \\
   --topology=cluster-slave \\
   --master-dataservice=west \\
   --slave-dataservice=east
```

h. Install the four services, that is, the dataservice service aliases `east` and `west` and the two services replicating between the clusters, `east_west` and `west_east`:

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

```shell
NOTE >> Configuration defaults updated in /home/tungsten/tungsten-replicator-2.2.0/deploy.cfg
NOTE >> Data service(s) east updated in /home/tungsten/tungsten-replicator-2.2.0/deploy.cfg
NOTE >> Data service(s) west updated in /home/tungsten/tungsten-replicator-2.2.0/deploy.cfg
NOTE >> Data service(s) east_west updated in /home/tungsten/tungsten-replicator-2.2.0/deploy.cfg
NOTE >> Data service(s) west_east updated in /home/tungsten/tungsten-replicator-2.2.0/deploy.cfg
```
Once all four services have been configured, the installation will output the status of each host.

Although four services are defined, once installed, they exist as only two distinct visible services:

- **east** service, operating on **west{1,2,3}** and reading from **east{1,2,3}**.
- **west** service, operating on **east{1,2,3}** and reading from **west{1,2,3}**.
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 the service starts correctly, you should see the configuration and current status of the service.

### 2.5.3. Management and Monitoring

Monitoring the dataservice is then divided into monitoring the two different clusters. Using `cctrl` gives you the dataservice status individually for the `east` and `west` dataservice. For example, the `east` dataservice is shown below:

```plaintext
Continuent Tungsten 2.0.5 build 3
east: session established
[LOGICAL] /east > ls
COORDINATOR[east1:AUTOMATIC:ONLINE]

Routers:
- connector@east1[17951](ONLINE, created=0, active=0)
- connector@east2[17939](ONLINE, created=0, active=0)
- connector@east3[17961](ONLINE, created=0, active=0)

DATASOURCES:
- east1(master:ONLINE, progress=29, THL latency=0.739)
  STATUS [OK] [2013/11/25 11:24:35 AM GMT]
  - MANAGER(state=ONLINE)
  - REPLICATOR(role=master, state=ONLINE)
  - DATASERVER(state=ONLINE)
  - CONNECTIONS(created=0, active=0)

- east2(slave:ONLINE, progress=29, latency=0.721)
  STATUS [OK] [2013/11/25 11:24:39 AM GMT]
  - MANAGER(state=ONLINE)
  - REPLICATOR(role=slave, master=east1, state=ONLINE)
  - DATASERVER(state=ONLINE)
  - CONNECTIONS(created=0, active=0)

- east3(slave:ONLINE, progress=29, latency=1.143)
  STATUS [OK] [2013/11/25 11:24:38 AM GMT]
  - MANAGER(state=ONLINE)
  - REPLICATOR(role=slave, master=east1, state=ONLINE)
  - DATASERVER(state=ONLINE)
  - CONNECTIONS(created=0, active=0)
```

When checking the current status, it is important to compare the sequence numbers from each service correctly. There are four services to monitor, the Continuent Tungsten service `east`, and a Tungsten Replicator service `east` that reads data from the `west` Continuent Tungsten service. A corresponding `west` Continuent Tungsten and `west` Tungsten Replicator service.

- When data is inserted on the master within the `east` Continuent Tungsten, use `cctrl` to determine the cluster status. Sequence numbers within the Continuent Tungsten `east` should match, and latency between hosts in the Continuent Tungsten service are relative to each other.

- When data is inserted on `east`, the sequence number of the `east` Continuent Tungsten service and `east` Tungsten Replicator service (on `west{1,2,3}`) should be compared.

- When data is inserted on the master within the `east` Continuent Tungsten, use `cctrl` to determine the cluster status. Sequence numbers within the Continuent Tungsten `east` should match, and latency between hosts in the Continuent Tungsten service are relative to each other.

- When data is inserted on `west`, the sequence number of the `west` Continuent Tungsten service and `west` Tungsten Replicator service (on `east{1,2,3}`) should be compared.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Continuent Tungsten Service Seqno</th>
<th>Tungsten Replicator Service Seqno</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insert/update data on <code>east</code></td>
<td>Seqno Increment</td>
<td>Seqno Increment</td>
</tr>
</tbody>
</table>
Within each cluster, `cctrl` can be used to monitor the current status. For more information on checking the status and controlling operations, see Section 4.1, "Checking Dataservice Status".

### Note

For convenience, the shell PATH can be updated with the tools and configuration. With two separate services, both environments must be updated. To update the shell with the Continuent Tungsten service and tools:

```
shell> source /opt/continuent/share/env.sh
```

To update the shell with the Tungsten Replicator service and tools:

```
shell> source /opt/replicator/share/env.sh
```

To monitor all services and the current status, you can also use the `multi_trepctl` command (part of the Tungsten Replicator installation). This generates a unified status report for all the hosts and services configured:

```
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>53</td>
<td>120.161</td>
</tr>
<tr>
<td>east2</td>
<td>east</td>
<td>slave</td>
<td>ONLINE</td>
<td>53</td>
<td>119.961</td>
</tr>
<tr>
<td>east3</td>
<td>east</td>
<td>master</td>
<td>ONLINE</td>
<td>53</td>
<td>119.834</td>
</tr>
<tr>
<td>west1</td>
<td>east</td>
<td>slave</td>
<td>ONLINE</td>
<td>53</td>
<td>181.128</td>
</tr>
<tr>
<td>west2</td>
<td>east</td>
<td>slave</td>
<td>ONLINE</td>
<td>53</td>
<td>204.790</td>
</tr>
<tr>
<td>west3</td>
<td>east</td>
<td>slave</td>
<td>ONLINE</td>
<td>53</td>
<td>22.895</td>
</tr>
<tr>
<td>east1</td>
<td>west</td>
<td>master</td>
<td>ONLINE</td>
<td>294327</td>
<td>0.285</td>
</tr>
<tr>
<td>east2</td>
<td>west</td>
<td>slave</td>
<td>ONLINE</td>
<td>294327</td>
<td>0.879</td>
</tr>
<tr>
<td>east3</td>
<td>west</td>
<td>slave</td>
<td>ONLINE</td>
<td>294327</td>
<td>0.567</td>
</tr>
<tr>
<td>west1</td>
<td>west</td>
<td>slave</td>
<td>ONLINE</td>
<td>231595</td>
<td>22.895</td>
</tr>
<tr>
<td>west2</td>
<td>west</td>
<td>master</td>
<td>ONLINE</td>
<td>231595</td>
<td>0.316</td>
</tr>
<tr>
<td>west3</td>
<td>west</td>
<td>slave</td>
<td>ONLINE</td>
<td>231595</td>
<td>1.046</td>
</tr>
</tbody>
</table>

In the above example, it can be seen that the `west` services have a much higher applied last sequence number than the `east` services, this is because all the writes have been applied within the `west` cluster.

To monitor individual servers and/or services, use `trepctl`, using the correct port number and servicename. For example, on `east1` to check the status of the replicator within the Continuent Tungsten service:

```
shell> trepctl status
```

To check the Tungsten Replicator service, explicitly specify the port and service:

```
shell> trepctl -port 10002 -service west status
```

For a number of standard operations, including basic monitoring, backup and restore, see Chapter 4, *Operations Guide*. For deployment specific operations, see the sections below.

### 2.5.3.1. Configuring Startup on Boot

Because there are two different Continuent services running, each must be individually configured to startup on boot:

- For the Continuent Tungsten service, use Section 2.13, "Configuring Startup on Boot".
- For the Tungsten Replicator service, a custom startup script must be created, otherwise the replicator will be unable to start as it has not been configured in a different directory.

1. Create a link from the Tungsten Replicator service startup script in the operating system startup directory (`/etc/init.d`):

```
shell> sudo ln -s /opt/replicator/tungsten/tungsten-replicator/bin/replicator /etc/init.d/mmreplicator
```

2. Modify the `APP_NAME` variable within the startup script (`/etc/init.d/mmreplicator`) to `mmreplicator`:

```
APP_NAME="mmreplicator"
```

3. Update the operating system startup configuration to use the updated script.

On Debian/Ubuntu:

```
shell> sudo update-rc.d mmreplicator defaults
```

---

### Deployment

<table>
<thead>
<tr>
<th>Operation</th>
<th>Continuent Tungsten Service Seqno</th>
<th>Tungsten Replicator Service Seqno</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insert/update data on</td>
<td><code>west</code> Seqno Increment</td>
<td><code>west</code> Seqno Increment</td>
</tr>
</tbody>
</table>

```
On RedHat/CentOS:

```
shell> sudo checkconfig --add mmreplicator
```

### 2.5.3.2. Resetting a single dataservice

Under certain conditions, dataservices in a multimaster configuration may drift and/or become inconsistent with the data in another dataservice. If this occurs, you may need to re-provision the data on one or more of the dataservices after first determining the definitive source of the information.

In the following example the `west` Service has been determined to be the definitive copy of the data. To fix the issue, all the datasources in the `east` service will be reprovisioned from one of the datasources in the `west` service.

The following is a guide to the steps that should be followed. In the example procedure it is the `east` service that has failed:

1. Put the dataservice into `MAINTENANCE` mode. This ensures that Continuent Tungsten will not attempt to automatically recover the service.
   ```
ctrl [east] > set policy maintenance
   ```
2. On the `east`, failed, Continuent Tungsten service, put each Tungsten Connector offline:
   ```
ctrl [east] > router * offline
   ```
3. Reset the failed Tungsten Replicator service on all servers connected to the failed Continuent Tungsten service. For example, on `west{1,2,3}` reset the `east` Tungsten Replicator service:
   ```
shell west> /opt/replicator/tungsten/tungsten-replicator/bin/trepctl -service east offline
shell west> /opt/replicator/tungsten/tungsten-replicator/bin/trepctl -service east reset -all -y
   ```
4. Reset the Continuent Tungsten service on each server within the failed region (`east{1,2,3}`):
   ```
shell east> /opt/continuent/tungsten/tungsten-replicator/bin/replicator stop
shell east> /opt/continuent/tungsten/tools/tpm reset east
shell east> /opt/continuent/tungsten/tungsten-replicator/bin/replicator start
   ```
5. Restore a backup on each host (`east{1,2,3}`) in the failed `east` service from a host in the `west` service:
   ```
shell east> /opt/continuent/tungsten/tungsten-replicator/scripts/tungsten_provision_slave --direct --source=west1
   ```
6. Place all the Tungsten Replicator services on `west{1,2,3}` back online:
   ```
shell west> /opt/replicator/tungsten/tungsten-replicator/bin/trepctl -service east online
   ```
7. On the `east`, failed, Continuent Tungsten service, put each Tungsten Connector online:
   ```
ctrl [east] > router * online
   ```
8. Set the policy back to automatic:
   ```
ctrl> set policy automatic
   ```

### 2.5.3.3. Resetting all dataservices

To reset all of the dataservices and restart the Continuent Tungsten and Tungsten Replicator services:

On all hosts (e.g. `east{1,2,3}` and `west{1,2,3}`):

```
shell> /opt/replicator/tungsten/tungsten-replicator/bin/replicator stop
shell> /opt/replicator/tungsten/tools/tpm reset
```

### 2.5.3.4. Provisioning during live operations

In the event of a failure within one host in the service where you need to reprovision the host from another running slave:

- Identify the servers that are failed. All servers that are not the master for their region can be re-provisioned using a backup/restore of the master (see Section 4.9, “Creating a Backup” or using the `tungsten_provision_slave` script.
- To re-provision an entire region, follow the steps below. The `east` region is used in the example statements below:
1. To prevent application servers from reading and writing to the failed service, place the Tungsten Connector offline within the failed region:

        cctrl [east]> router * offline

2. On all servers in other regions (west{1,2,3}):

        shell> /opt/replicator/tungsten/tungsten-replicator/bin/trepctl -service east offline
        shell> /opt/replicator/tungsten/tungsten-replicator/bin/trepctl -service east reset -all -y

3. On all servers in the failed region (east{1,2,3}):

        shell> /opt/replicator/tungsten/tungsten-replicator/bin/replicator stop
        shell> /opt/continuent/tungsten/tungsten-replicator/tools/tpm reset
        shell> /opt/continuent/tungsten/tungsten-replicator/scripts/tungsten_provision_slave
                      --direct --source=west1

4. Check that Continuent Tungsten is working correctly and all hosts are up to date:

        cctrl [east]> ls

5. Restart the Tungsten Replicator service:

        shell> /opt/replicator/tungsten/tungsten-replicator/bin/replicator start

6. On all servers in other regions (west{1,2,3}):

        shell> /opt/replicator/tungsten/tungsten-replicator/bin/trepctl -service east online

2.5.3.5. Dataserver maintenance

To perform maintenance on the dataservice, for example to update the MySQL configuration file, can be achieved in a similar sequence to that shown in Section 4.11, "Performing Database or OS Maintenance", except that you must also restart the corresponding Tungsten Replicator service after the main Continuent Tungsten service has been placed back online.

For example, to perform maintenance on the east service:

1. Put the dataservice into `MAINTENANCE` mode. This ensures that Continuent Tungsten will not attempt to automatically recover the service.

        cctrl [east]> set policy maintenance

2. Shun the first slave datasource so that maintenance can be performed on the host.

        cctrl [east]> datasource east1 shun

3. Perform the updates, such as updating `my.cnf`, changing schemas, or performing other maintenance.

4. If MySQL configuration has been modified, restart the MySQL service:

        cctrl [east]> service host/mysql restart

5. Bring the host back into the dataservice:

        cctrl [east]> datasource host recover

6. Perform a switch so that the master becomes a slave and can then be shunned and have the necessary maintenance performed:

        cctrl [east]> switch

7. Repeat the previous steps to shun the host, perform maintenance, and then switch again until all the hosts have been updated.

8. Set the policy back to automatic:

        cctrl> set policy automatic

9. On each host in the other region, manually restart the Tungsten Replicator service, which will have gone offline when MySQL was restarted:

        shell> /opt/replicator/tungsten/tungsten-replicator/bin/trepctl -host host --service east online

2.5.3.5.1. Fixing Replication Errors

In the event of a replication fault, the standard `cctrl`, `trepctl` and other utility commands in Chapter 7, Command-line Tools can be used to bring the dataservice back into operation. All the tools are safe to use.
If you have to perform any updates or modifications to the stored MySQL data, ensure binary logging has been disabled using:

```sql
mysql> SET SESSION SQL_LOG_BIN=0;
```

Before running any commands. This prevents statements and operations reaching the binary log so that the operations will not be replicated to other hosts.

### 2.6. Deploying Composite (SOR) Topologies

Continuent Tungsten supports the creation of composite, or Statement of Record (SOR) topologies.

Figure 2.5. Topologies: Composite

```bash
./tools/tpm configure defaults
   --user=tungsten
   --install-directory=/opt/continuent
   --mysql-connector-j-path=/opt/mysql/mysql-connector-java-5.1.18/mysql-connector-java-5.1.18-bin.jar
   -- datasource-user=tungsten
   -- datasource-password=secret
   -- application-user=application_user
   -- application-password=application_password
   -- connector-listen-port=3306
   -- datasource-port=3306
   -- datasource-log-directory=/var/lib/mysql
   -- profile-script-"~/.bashrc"
   -- start-and-report
./tools/tpm configure nyc
   --dataservice-hosts=db1.nyc.example.com,db2.nyc.example.com,db3.nyc.example.com
   --dataservice-connectors=db1.nyc.example.com,db2.nyc.example.com,db3.nyc.example.com,conn1.nyc.example.com
   --dataservice-master-host=db1.nyc.example.com
   --dataservice-witnesses=router.nyc.example.com
./tools/tpm configure sjc
   --dataservice-relay-enabled=true
   --dataservice-relay-source=nyc
   --dataservice-hosts=db1.sjc.example.com,db2.sjc.example.com,db3.sjc.example.com
```
We can identify the different elements in the above command as follows:

- **./tools/tpm install dsone**
  
  This runs the `tpm` command. `install` indicates that we are installing and creating a new dataservice, and `dsone` is the name and identity of the dataservice being created.

- **--user=tungsten**
  
  The operating system user name that you have created for the Tungsten service, `tungsten`.

- **--install-directory=/opt/continuent**
  
  The installation directory of the Tungsten service. This is where the service will be installed on each server in your dataservice.

- **--members=cont-db1,cont-db2,cont-db3**
  
  A comma separated list of all the hosts that are part of this dataservice.

- **--connectors=cont-db1,cont-db2,cont-db3**
  
  A comma separated list of the hosts that will have a connector service created on them.

- **--master=cont-db1**
  
  The hostname of the server that will be the master MySQL server.

- **--dataservice-witnesses=witness**
  
  The hostname of a computer that will be contacted using `ping` in the event of a network problem.

- **--mysql-connectorj-path=/usr/share/java/mysql-connector-java-5.1.16.jar**
  
  The location of the JAR file for the MySQL Connector/J component. This should be the location of the JAR, not a symbolic link.

- **--datasource-user=tungsten**
  
  The MySQL user name to use when connecting to the MySQL database.

- **--datasource-password=password**
  
  The MySQL password for the user that will connect to the MySQL database.

- **--application-user=app_user**
  
  The application user name.

- **--application-password=password**
  
  The application password.

- **--datasource-port=13306**
  
  The TCP/IP port that the MySQL database is listening on for connections.

- **--connector-listen-port=3306**
  
  The TCP/IP port on which to listen for incoming connections by the Tungsten connector service. To emulate the standard MySQL database service, port 3306 is used.

- **--datasource-log-directory=/var/log/mysql**
  
  The location of the binary log files for MySQL. The above directory is the default.
The profile script used when your shell starts. Using this line modifies your profile script to add a path to the Tungsten tools so that managing Continuent Tungsten™ are easier to use.

- `--start-and-report`
  
  Tells Tungsten to startup the service, and report the current configuration and status.

### 2.7. Replicating Data Out of an Existing 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.

The deployment works by creating an alias into the Continuent Tungsten cluster, and then uses this as the source information for configuring a separate service from the alias into the standalone replicator.

**Figure 2.6. Topologies: Master/Slave with Replicator**

In order to configure this, there are three steps:

1. Setup a new server to hold the Tungsten Replicator service
2. Configure an alias to the existing Continuent Tungsten service
3. Configure a replicator that points to the alias acting as a slave

There are also the following requirements:
• The host on which you want to replicate to must have Tungsten Replicator 2.1.2 or later
• Hosts on both the replicator and cluster must be able to communicate with each other.
• Replicator must be able to connect as the **tungsten** user to the databases within the cluster

The **tpm** command to create the service on the replicator is as follows:

1. On your staging server, download the release package.
2. Unpack the release package:
   ```
   $ tar zxf tungsten-replicator-3.0.0-500.tar.gz
   ```
3. Change to the unpackaged directory:
   ```
   $ cd tungsten-replicator-3.0.0-500
   ```
4. Execute the **tpm** command to configure the installation:
   ```
   $ ./tools/tpm configure defaults \
   --install-directory=/opt/replicator \
   --profile-script=~/.bashrc \
   --replication-password=secret \
   --replication-port=13306 \
   --replication-user=tungsten \
   --start-and-report=true \
   --user=tungsten
   ```

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 configure**
  Configures default options that will be configured for all future services.
  - **--install-directory=/opt/replicator** [234]
    The installation directory of the Tungsten service. This is where the service will be installed on each server in your dataservice.
  - **--profile-script="~/.bashrc"** [244]
    The profile script used when your shell starts. Using this line modifies your profile script to add a path to the Tungsten tools so that managing Continuent Tungsten™ are easier to use.
  - **--user=tungsten** [251]
    The operating system user name that you have created for the Tungsten service, **tungsten**.
  - **--replication-user=tungsten** [246]
    The user name that will be used to apply replication changes to the database on slaves.
  - **--replication-password=password** [245]
    The password that will be used to apply replication changes to the database on slaves.
  - **--replication-port=13306** [246]
    Set the port number to use when connecting to the MySQL server.
  - **--start-and-report** [248]
    Tells **tpm** to startup the service, and report the current configuration and status.

5. Now the defaults are configure, first we configure a cluster alias that points to the masters and slaves within the current Continuent Tungsten service that you are replicating from:
   ```
   $ ./tools/tpm configure beta \
   --master=hostA \ 
   --slaves=hostB \ 
   --topology=cluster-alias
   ```

   This creates a configuration that specifies that the topology should be a cluster alias, and that the cluster configuration of the cluster from which you are replicating, the master is **hostA** and the slave is **hostB**.
6. Now, the service that will replicate from the cluster alias into the database on hostC can be created:

```bash
shell> /tools/tpm configure gamma
   --master=hostC
   --relay-source=beta
   --topology=cluster-slave
```

This creates the configuration that will replicate into the database on the Tungsten Replicator. In this example:

- `--master=hostC` [237]
  Specifies the hostname of the database into which data will be replicated.

- `--relay-source=beta` [245]
  Specifies the name of the relay source for data that will be used to read events to be replicated.

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

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

The cluster should be installed and ready to use.

### 2.8. 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 cluster through a connector attached to your dataservice. By writing through the connector, changes to the underlying dataservice topology can be handled.

Figure 2.7. Topologies: Replicating into a Dataservice

In order to configure this deployment, there are two steps:
1. Create a new replicator on an existing server that replicates into a connector
2. Create a new replicator that reads the binary logs directly from the external MySQL service through the connector

There are also the following requirements:

- The host on which you want to replicate to must have Tungsten Replicator 2.2.0 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 `GRANT RELOAD, REPLICATION SLAVE, and REPLICATION CLIENT` privileges.
- Replicator must be able to connect as the `tungsten` user to the databases within the cluster.
- When writing into the master through the connector, the user must be given the correct privileges to write and update the MySQL server. For this reason, the easiest method is to use the `tungsten` user, and ensure that that user has been added to the `user.map`:

```
tungsten secret alpha
```

The `tpm` command to create the service on the replicator should be executed on `host1`, after the Tungsten Replicator distribution has been extracted:

```
shell> cd tungsten-replicator-2.2.1
shell> ./tools/tpm configure defaults
--install-directory=/opt/replicator
--rmi-port=10002
--user=tungsten
--replication-user=tungsten
--replication-password=secret
--replication-port=3306
--direct-replication-port=13306
--skip-validation-check=MySQLApplierPortCheck
--skip-validation-check=MySQLNoMySQLReplicationCheck
--log-slave-updates=true
```

This configures the default configuration values that will be used for the replication service.

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

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

- **tpm configure**

  Configures default options that will be configured for all future services.

  **--install-directory=/opt/continuent [234]**

  The installation directory of the Tungsten service. This is where the service will be installed on each server in your dataservice.

  **--rmi-port=10002 [246]**

  Configure a different RMI port from the default selection to ensure that the two replicators do not interfere with each other.

  **--user=tungsten [251]**

  The operating system user name that you have created for the Tungsten service, `tungsten`.

  **--replication-user=tungsten [246]**

  The user name that will be used to apply replication changes to the database on slaves.

  **--replication-password=password [245]**

  The password that will be used to apply replication changes to the database on slaves.

  **--replication-port=3306 [246]**

  Set the port number to use when connecting to the MySQL server.

  **--direct-replication-port=13306**

  Set the port number to use when writing data to the MySQL server through the connector.

Now that the defaults are configured, first we configure a cluster alias that points to the masters and slaves within the current Continuent Tungsten service that you are replicating from:
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
```

Once the installation has been completed, you must update the position of the replicator so that it points to the correct position within the source database to prevent errors during replication. If the replication is being created as part of a migration process, determine the position of the binary log from the external replicator service used when the backup was taken. For example:

```
mysql> show master status;
```

```
*************************** 1. row ***************************
    File: mysql-bin.000026
  Position: 1311
Binlog_Do_DB: 
Binlog_Ignore_DB: 
1 row in set (0.00 sec)
```

Use `tungsten_set_position` to update the replicator position to point to the master log position:

```
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:

```
shell> /opt/replicator/tungsten/tungsten-replicator/bin/replicator start
```

Replication status should be checked by explicitly using the servicename and/or RMI port:

```
shell> /opt/replicator/tungsten/tungsten-replicator/bin/trepctl status Processing status command...
NAME       VALUE
------ ------
appliedLastEventId : mysql-bin.000026:00000000000000001311;1252
appliedLastSeqno  : 5
appliedLatency    : 0.748
channels          : 1
clusterName       : beta
currentEventId    : mysql-bin.000026:00000000000000001311
currentTimeMillis : 135041681881
dataServerHost    : host1
extensions        :
host              :
latestEpochNumber : 1
masterConnectUri  : thl://host3:2112/
masterListenUri   : thl://host1:2113/
maximumStoredSeqNo: 5
minimumStoredSeqNo: 0
offlineRequests   : NONE
pendingError      : NONE
pendingErrorCode  : NONE
pendingErrorEventId: NONE
pendingErrorSeqno : -1
pendingExceptionMessage: NONE
pipelineSource    : jdbc:mysql:thin://host3:13306/
relativeLatency   : 8408.881
resourcePrecedence: 99
rmiPort           : 10000
role              : master
seqnoType         : java.lang.Long
serviceName       : beta
serviceType       : local
simpleServiceName : beta
siteName          : default
sourceId          : host3
state             : ONLINE
timeInStateSeconds: 8409.21
transitioningTo   : 
uptimeSeconds     : 8409.88
useSSLConnection  : false
version           : Tungsten Replicator 2.0.5 build 3
Finished status command...
```
2.9. Deploying Additional Datasources, Managers, or Connectors

2.9.1. Adding Datasources to an Existing Deployment

1. Ensure the new host that is being added has been configured following the Appendix C, Prerequisites.

2. Update the configuration using `tpm`, adding the new host to the list of `--members` and `--hosts` using `+=`, which appends the host to the Existing Deployment:

```
shell> ./tools/tpm update alpha --members+=host6 --hosts+=host6
```

If the host will also act as a connector, also add it to the list of connectors:

```
shell> ./tools/tpm update alpha --members+=host6 --hosts=host6 --connectors+=host6
```

3. Initially, the newly added host will attempt to read the information from the existing THL. If the full THL is not available from the master, the new slave will need to be reprovisioned:

   a. Log into the new host.

   b. Execute `tungsten_provision_slave` to read the information from an existing existing and overwrite the data within the new host:

```
shell> tungsten_provision_slave --source=host2
```

**NOTE** >> Put alpha replication service offline

**NOTE** >> Create a mysqldump backup of host2 in `/opt/continuent/backups/provision_mysqldump_2014-01-17_17-27_96`

**NOTE** >> host2 >> Create mysqldump in `/opt/continuent/backups/provision_mysqldump_2014-01-17_17-27_96/provision.sql.gz`

**NOTE** >> Load the mysqldump file

**NOTE** >> Put the alpha replication service online

**NOTE** >> Clear THL and relay logs for the alpha replication service

Once the new host has been added and re-provision, check the status in `cctrl`:

```
(LOGICAL) /alpha > ls

COORDINATOR[ host1:AUTOMATIC:ONLINE ]

Routers:

  |connector@host1[11401](ONLINE, created=0, active=0)
  |connector@host2[8756](ONLINE, created=0, active=0)
  |connector@host3[21673](ONLINE, created=0, active=0)

Datarources:

  |host1(master:ONLINE, progress=219, THL latency=1.047) [STATUS [OK] [2013/12/13 04:16:17 PM GMT]]
  |  MANAGER(state=ONLINE)  
  |  REPLICATOR(role=master, state=ONLINE)  
  |  DATASERVER(state=ONLINE)
  |  CONNECTIONS(created=0, active=0)

  |host2(slave:ONLINE, progress=219, latency=1.588) [STATUS [OK] [2013/12/13 04:16:17 PM GMT]]
  |  MANAGER(state=ONLINE)
  |  REPLICATOR(role=slave, master=host1, state=ONLINE)
  |  DATASERVER(state=ONLINE)
  |  CONNECTIONS(created=0, active=0)

  |host3(slave:ONLINE, progress=219, latency=2.021) [STATUS [OK] [2013/12/13 04:16:18 PM GMT]]
  |  MANAGER(state=ONLINE)
  |  REPLICATOR(role=slave, master=host1, state=ONLINE)
  |  DATASERVER(state=ONLINE)
  |  CONNECTIONS(created=0, active=0)

  |host6(slave:ONLINE, progress=219, latency=1.000) [STATUS [OK] [2014/01/17 03:28:54 PM GMT]]
```
If the host has not come up, or the progress does not match the master, check Section 4.5, "Datasource Recovery Steps" for more information on determining the exact status and what steps to take to enable the host operation.

### 2.9.2. Adding Active Witnesses to an Existing Deployment

To add active witnesses to an Existing Deployment, use `tpm` to update the configuration, adding the list of active witnesses and the list of all members within the updated dataservice configuration.

Active Witness hosts must have been prepared using the notes provided in Appendix C, Prerequisites. Active witnesses must be able to resolve the hostnames of the other managers and hosts in the dataservice. Installation will fail if prerequisites and host availability and stability cannot be confirmed.

To update the configuration:

```shell
./tools/tpm update alpha --witnesses=host4 --members+host4 --enable-active-witnesses=true
```

Once installation has completed successfully, and the manager service has started on each configured active witness, the status can be determined using `ls` within `cctrl`:

```
[LOGICAL] /alpha > ls
COORDINATOR[host1:AUTOMATIC:ONLINE]
ROUTERS:
  connector@host1[20446](ONLINE, created=0, active=0)
  connector@host2[21698](ONLINE, created=0, active=0)
  connector@host3[30354](ONLINE, created=0, active=0)
DATASOURCES:
  host1(slave:ONLINE, progress=8946, latency=0.000)
    STATUS [OK] [2013/12/05 04:27:47 PM GMT]
    MANAGER(state=ONLINE)
    REPLICA(role=slave, master=host3, state=ONLINE)
    DATASERVER(state=ONLINE)
    CONNECTIONS(created=0, active=0)

  host2(slave:ONLINE, progress=8946, latency=0.334)
    STATUS [OK] [2013/12/05 04:06:59 PM GMT]
    MANAGER(state=ONLINE)
    REPLICA(role=slave, master=host3, state=ONLINE)
    DATASERVER(state=ONLINE)
    CONNECTIONS(created=0, active=0)

  host3(master:ONLINE, progress=8946, THL latency=0.331)
    STATUS [OK] [2013/11/20 05:39:14 PM GMT]
    MANAGER(state=ONLINE)
    REPLICA(role=master, state=ONLINE)
    DATASERVER(state=ONLINE)
    CONNECTIONS(created=0, active=0)

WITNESSES:
  host4(witness:ONLINE)
```

Validation of the cluster with the new witnesses can be verified by using the `cluster validate` command within `cctrl`.

### 2.9.3. Adding Passive Witnesses to an Existing Deployment

To add passive witness to an existing installation, use `tpm` to update the active configuration.
Continuent recommend that active witnesses, rather than passive witnesses are used for all installations. For more information on differences, see Section 2.1, “Host Types”.

Once the update process has been completed, the current status will be displayed using `cctrl`. The current configuration of the cluster can be verified by using the `cluster validate` command within `cctrl`.

2.9.4. Adding Connectors to an Existing Deployment

Adding more connectors to an existing installation allows for increased routing capacity. The new connectors will form part of the cluster and be fully aware and communicate with existing managers and datasources within the cluster.

To add more connectors to an existing deployment:

1. On the new host, ensure the Appendix C, Prerequisites have been followed.

2. On the staging host, update the configuration to include the connector host you are adding. For example, to update the list of connector hosts to include `host4` on the service `alpha`:

```shell
./tools/tpm configure alpha --connectors+=host4
```

   The `+=` appends the new host to the list of supported hosts.

3. Update the configuration, which will install the connector on the new host:

```shell
./tools/tpm update --no-connectors
```

   Using the `--no-connectors` option updates the current deployment without restarting the existing connectors.

4. During a period when it is safe to restart the connectors:

```shell
./tools/tpm promote-connector
```

   The status of all the connectors can be monitored using `cctrl`:

```
[LOGICAL] /alpha > ls
COORDINATOR[host1:AUTOMATIC:ONLINE]
ROUTERS:
  [connector@host1[8616](ONLINE, created=0, active=0)]
  [connector@host2[12381](ONLINE, created=0, active=0)]
  [connector@host3[19708](ONLINE, created=0, active=0)]
  [connector@host4[5085](ONLINE, created=0, active=0)]
```

2.9.5. Adding a remote Composite Cluster

2.10. Removing Datasources, Managers or Connectors

Removing components from a dataservice is quite straightforward, usually involved 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 2.10.1, “Removing a Datasource from an Existing Deployment”
• Section 2.10.2, "Removing a Connector from an Existing Deployment"

### 2.10.1. 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. Check the current service state:

   ```
   [LOGICAL] /alpha > ls
   COORDINATOR[host1:AUTOMATIC:ONLINE]
   ROUTERS:
   | connector@host1[11401](ONLINE, created=17, active=0) |
   | connector@host2[7998](ONLINE, created=0, active=0) |
   | connector@host3[31540](ONLINE, created=0, active=0) |
   | connector@host4[26829](ONLINE, created=27, active=1) |
   +----------------------------------------------------------------------------+
   DATASOURCES:
   | host1(slave:ONLINE, progress=373, latency=0.000) |
   | STATUS [OK] [2014/02/12 12:48:14 PM GMT] |
   |  MANAGER(state=ONLINE) |
   |  REPLICATOR(role=slave, master=host6, state=ONLINE) |
   |  DATASERVER(state=ONLINE) |
   |  CONNECTIONS(created=30, active=0) |
   +----------------------------------------------------------------------------+
   | host2(slave:ONLINE, progress=373, latency=1.000) |
   | STATUS [OK] [2014/01/24 05:02:34 PM GMT] |
   |  MANAGER(state=ONLINE) |
   |  REPLICATOR(role=slave, master=host6, state=ONLINE) |
   |  DATASERVER(state=ONLINE) |
   |  CONNECTIONS(created=0, active=0) |
   +----------------------------------------------------------------------------+
   | host3(slave:ONLINE, progress=373, latency=1.000) |
   | STATUS [OK] [2014/02/11 03:17:08 PM GMT] |
   |  MANAGER(state=ONLINE) |
   |  REPLICATOR(role=slave, master=host6, state=ONLINE) |
   |  DATASERVER(state=ONLINE) |
   |  CONNECTIONS(created=0, active=0) |
   +----------------------------------------------------------------------------+
   | host6(master:ONLINE, progress=373, THL latency=0.936) |
   | STATUS [OK] [2014/02/12 12:39:52 PM GMT] |
   |  MANAGER(state=ONLINE) |
   |  REPLICATOR(role=master, state=ONLINE) |
   |  DATASERVER(state=ONLINE) |
   |  CONNECTIONS(created=14, active=1) |
   +----------------------------------------------------------------------------+
   ```

2. Switch to **MAINTENANCE** policy mode:

   ```
   [LOGICAL] /alpha > set policy maintenance
   policy mode is now MAINTENANCE
   ```

3. Switch to administration mode:

   ```
   [LOGICAL] /alpha > admin
   ```

4. Remove the node from the active service using the `rm` command. You will be warned that this is an expert command and to confirm the operation:

   ```
   [ADMIN] /alpha > rm host6
   WARNING: This is an expert-level command:
   Incorrect use may cause data corruption
   ```
5. Switch back to logical mode:

```
[ADMIN] /alpha > logical
```

6. Switch to AUTOMATIC policy mode:

```
[LOGICAL] /alpha > set policy automatic
policy mode is now AUTOMATIC
```

Now the node has been removed from the active dataservice, the services must be stopped and then removed from the configuration.

1. Stop the running services:

```
shell> stopall
```

2. Now you must remove the node from the configuration, although the exact method depends on which installation method used with `tpm`:
   - If you are using staging directory method with `tpm`:
     a. Change to the staging directory. The current staging directory can be located using `tpm query staging`:
        ```
        shell> tpm query staging
tungsten@host1:/home/tungsten/continuent-tungsten-2.0.5-3
        shell> cd /home/tungsten/continuent-tungsten-2.0.5-3
        ```
     b. Update the configuration, omitting the host from the list of members of the dataservice:
        ```
        shell> tpm update alpha
        --connectors=host1,host2,host3,host4
        --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
        ```
  3. Stop the replicator/manager from being started again.
    - If this all the services on the this node, replicator, manager and connector are being removed, remove the Continuent Tungsten installation entirely:
      a. Remove the startup scripts from your server:
         ```
         shell> sudo /opt/continuent/tungsten/cluster-home/bin/undeployall
         ```
      b. Remove the installation directory:
         ```
         shell> rm -rf /opt/continuent
         ```
    - If the replicator/manager has been installed on a host but the connector is not being removed, remove the start scripts to prevent the services from being automatically started:
      ```
      shell> rm /etc/init.d/tmanager
      shell> rm /etc/init.d/treplicator
      ```

2.10.2. Removing a Connector from an Existing Deployment

Removing a connector involves only stopping the connector and removing the configuration. When the connector is stopped, the manager will automatically remove it from the dataservice. Note that applications that have been configured to talk to the connector must be updated to point to another connector.

For example, to remove host4 from the current dataservice:

1. Login to the host running the connector.
2. Stop the connector service:

```
shell> connector stop
```
3. Remove the connector from the configuration, the exact method depends on which installation method used with tpm:
   • If you are using staging directory method with tpm:
     a. Change to the staging directory. The current staging directory can be located using tpm query staging:

```
shell> tpm query staging
```

```
tungsten@host1:/home/tungsten/continuent-tungsten-2.0.5-3
shell> cd /home/tungsten/continuent-tungsten-2.0.5-3
```

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

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

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

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

4. Stop the connector from being started again. If the connector is restarted, it will connect to the previously configured masters and begin operating again.
   • If this is a standalone Connector installation, remove the Continuent Tungsten installation entirely:
     • Remove the startup scripts from your server:

```
shell> sudo /opt/continuent/tungsten/cluster-home/bin/undeployall
```

• Remove the installation directory:

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

• If the connector has been installed on a host with replicator and/or managers, remove the start script to prevent the connector from being automatically started:

```
shell> rm /etc/init.d/tconnector
```

### 2.11. Deploying Tungsten Connector Only

An independent Tungsten Connector installation can be useful when you want to create a connector service that provides HA and load balancing, but which operates independently of the main cluster. Specifically, this solution is used within disaster recovery and multi-site operations where the connector may be operating across site-boundaries independently of the dataservice at each site.

The independent nature is in terms of the configuration of the overall service through tpm; an independent connector configured to communicate with existing cluster hosts will be managed by the managers of the cluster. But, the connector will not be updated when performing a tpm update operation within the configured cluster. This allows the connector to work through upgrade procedures to minimize downtime.

To create an independent connector, tpm is used to create a definition for a cluster including the datasources, and specifying only a single connector host, then installing Continuent Tungsten on only the connector host. Failure to configure in this way, and tpm will install a full Continuent Tungsten service across all the implied members of the cluster.

Installation can be configured using the staging directory method of tpm as follows:

1. On your staging server, download the release package.

2. Unpack the release package:

```
shell> tar zxf continuent-tungsten-2.0.5-3.tar.gz
```

3. Change to the unpackaged directory:

```
shell> cd continuent-tungsten-2.0.5-3
```

4. Create a cluster definition using tpm configure:

```
shell> /tools/tpm configure alpha
  --user=tungsten
  --install-directory=/opt/continuent
  --replication-port=13306
  --application-port=3306
  --application-user=app_user
  --application-password=password
```
### Deployment

```
--mysql-connectorj-path=/usr/share/java/mysql-connector-java-bin.jar
--members=host1,host2,host3
--connectors=`hostname`
```

The above creates a configuration specifying the datasources, `host{1,2,3}`, and a single connector host based on the hostname of the installation host. Note that the application and datasource port configuration are the same as required by a typical Continuent Tungsten configuration. The values above are identical to those used in Section 2.4, "Deploying a Master/Slave Topology" deployment.

5. Now execute the installation, only installing the service on the current host, again using `hostname`:

```
shell> ./tools/tpm install --hosts=`hostname`
```

### Next Steps

Unless automatically started, you must start the Tungsten services before the cluster will be available. Use the `tpm` command to start the services:

```
tools/tpm start --hosts=connector2
```

Wait a minute for the services to start up and configure themselves. After that you may proceed.

Once your services start successfully you may begin to use the cluster. To look at services and perform administration, run the following command from any database server:

```
$CONTINUENT_ROOT/tungsten/tungsten-manager/bin/cctrl
```

Configuration is now complete. For further information, please consult Tungsten documentation, which is available at docs.continuent.com.

NOTE >> Command successfully completed

6. Start the connector service:

```
shell> ./tools/tpm start alpha --hosts=connector
```

When using the INI file method of deployment, create a file containing the list of members, listing the current host as the only connector:

1. Download and extract the Continuent Tungsten distribution, and change into the distribution directory. See Section 2.3, "Deployment Sources" for more information.

2. Create the INI file `/etc/tungsten/tungsten.ini` with the following contents:

   ```ini
   [defaults]
   application-password=password
   application-user=app_user
   install-directory=/opt/continuent
   replication-port=13306
   user=tungsten
   
   [alpha]
   connectors=connector
   master=host1
   members=host2,host2,host3
   ```

3. Install the connector:

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

4. Start the connector:

```
shell> ./tools/tpm start
```

Once started:

- The connector will appear, and be managed by, any manager host using the `cctrl` tool. For example:

```
[LOGICAL] /dsone > ls
COORDINATOR[host1:AUTOMATIC:ONLINE]

ROUTERS:

| connector@connector2[16019] (ONLINE, created=0, active=0) |
| connector@host1[18450] (ONLINE, created=19638, active=0) |
| connector@host2[18450] (ONLINE, created=0, active=0) |
| connector@host3[8895] (ONLINE, created=0, active=0) |
```
• The active status of the connector can be monitored using \texttt{cctrl} as normal.

• Updates to the main cluster will not update the Continuent Tungsten of the standalone connector. The standalone must be updated independently of the remainder of the Continuent Tungsten dataservice.

• Connector can be accessed using the connector host and specified port:

  \begin{verbatim}
  shell> mysql -utungsten -p -hconnector -P3306
  \end{verbatim}

• The user.map authorization file must be created and managed separately on standalone connectors. For more information, see Section 5.4, “User Authentication”

### 2.12. Starting and Stopping Continuent Tungsten

To stop all of the services associated with a dataservice node, use the \texttt{stopall} script:

\begin{verbatim}
shell> stopall
Stopping Tungsten Connector...
Stopped Tungsten Connector.
Stopping Tungsten Replicator Service...
Stopped Tungsten Replicator Service.
Stopping Tungsten Manager Service...
Stopped Tungsten Manager Service.
\end{verbatim}

To start all services, use the \texttt{startall} script:

\begin{verbatim}
shell> startall
Starting Tungsten Manager Service...
Starting Tungsten Replicator Service...
Starting Tungsten Connector...
\end{verbatim}

#### 2.12.1. Restarting the Replicator Service

\textbf{Warning}

Restarting a running replicator temporarily stops and restarts replication. Either set \texttt{maintenance} mode within \texttt{cctrl} (see Section 4.11, “Performing Database or OS Maintenance” or shun the datasource before restarting the replicator (Section 4.1.5.1, “Shunning a Datasource”.

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

\begin{verbatim}
shell> replicator stop
Stopping Tungsten Replicator Service...
Stopped Tungsten Replicator Service.
\end{verbatim}

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

\begin{verbatim}
shell> replicator start
Starting Tungsten Replicator Service...
\end{verbatim}

#### 2.12.2. Restarting the Connector Service

\textbf{Warning}

Restarting the connector service will interrupt the communication of any running application or client connecting through the connector to MySQL.

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

\begin{verbatim}
shell> connector stop
Stopping Tungsten Connector Service...
Stopped Tungsten Connector Service.
\end{verbatim}

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

\begin{verbatim}
shell> connector start
Starting Tungsten Connector Service...
Waiting for Tungsten Connector Service....
running: PID 12348
\end{verbatim}
If the cluster was configured with `auto-enable=false` then you will need to put each node online individually.

### 2.12.3. Restarting the Manager Service

The manager service is designed to monitor the status and operation of each of the datasource within the dataservice. In the event that the manager has become confused with the current configuration, for example due to a network or node failure, the managers can be restarted. This forces the managers to update their current status and topology information.

Before restarting managers, the dataservice should be placed in maintenance policy mode. In maintenance mode, the connectors will continue to service requests and the manager restart will not be treated as a failure.

To restart the managers across an entire dataservice, each manager will need to be restarted. The dataservice must be placed in maintenance policy mode first, then:

1. To set the maintenance policy mode:
   ```
   [LOGICAL:EXPERT] /dsone > set policy maintenance
   ```

2. On each datasource in the dataservice:
   a. Stop the service:
     ```
     shell> manager stop
     ```
   b. Then start the manager service:
     ```
     shell> manager start
     ```

3. Once all the managers have been restarted, set the policy mode back to the automatic:
   ```
   [LOGICAL:EXPORT] /alpha > set policy automatic
   policy mode is now AUTOMATIC
   ```

### 2.13. Configuring Startup on Boot

By default, Continuent Tungsten does not start automatically on boot. To enable Continuent Tungsten to start at boot time, use the `deployall` script provided in the installation directory to create the necessary boot scripts:

```
shell> sudo /opt/continuent/tungsten/cluster-home/bin/deployall
Adding system startup for /etc/init.d/tmanager ... 
/etc/rc0.d/K80tmanager -> ../init.d/tmanager
/etc/rc1.d/K80tmanager -> ../init.d/tmanager
/etc/rc2.d/S80tmanager -> ../init.d/tmanager
/etc/rc3.d/S80tmanager -> ../init.d/tmanager
/etc/rc4.d/S80tmanager -> ../init.d/tmanager
/etc/rc5.d/S80tmanager -> ../init.d/tmanager
Adding system startup for /etc/init.d/treplicator ... 
/etc/rc0.d/K81treplicator -> ../init.d/treplicator
/etc/rc1.d/K81treplicator -> ../init.d/treplicator
/etc/rc2.d/S81treplicator -> ../init.d/treplicator
/etc/rc3.d/S81treplicator -> ../init.d/treplicator
/etc/rc4.d/S81treplicator -> ../init.d/treplicator
/etc/rc5.d/S81treplicator -> ../init.d/treplicator
Adding system startup for /etc/init.d/tconnector ... 
/etc/rc0.d/K82tconnector -> ../init.d/tconnector
/etc/rc1.d/K82tconnector -> ../init.d/tconnector
/etc/rc2.d/S82tconnector -> ../init.d/tconnector
/etc/rc3.d/S82tconnector -> ../init.d/tconnector
/etc/rc4.d/S82tconnector -> ../init.d/tconnector
/etc/rc5.d/S82tconnector -> ../init.d/tconnector
```

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

```
shell> sudo /opt/continuent/tungsten/cluster-home/bin/undeployall
```

### 2.14. Upgrading Continuent Tungsten

To upgrade an existing installation on Continuent Tungsten, 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 cluster as the updated versions of the tools are restarted, but downtime is deliberately kept to a minimum, and the cluster should be in the same operation state once the upgrade has finished as it was when the upgrade was started.
2.14.1. Upgrading using the Staging Method (with ssh Access)

Note

For INI file upgrades, see Section 2.14.2, “Upgrading when using INI-based configuration, or without ssh Access”

To perform an upgrade of an entire cluster from a staging directory installation, where you have ssh access to the other hosts in the cluster:

1. On your staging server, download the release package.
2. Unpack the release package:
   ```shell
   tar zxf continuent-tungsten-2.0.5-3.tar.gz
   ```
3. Change to the unpackaged directory:
   ```shell
   cd continuent-tungsten-2.0.5-3
   ```
4. The next step depends on your existing deployment:
   - **If you are upgrading a Multi-Site, Multi-Master deployment:**
     If you installed the original service by making use of the $CONTINUENT_PROFILES and $REPLICATOR_PROFILES environment variables, no further action needs to be taken to update the configuration information. Confirm that these variables are set before performing the validation and update.
     
     If you did not use these environment variables when deploying the solution, you must load the existing configuration from the current hosts in the cluster before continuing by using tpm fetch:
     ```shell
     ./tools/tpm fetch --hosts=east1,east2,east3,west1,west2,west3 \
     --user=tungsten --directory=/opt/continuent
     ```
     **Important**
     You must specify ALL the hosts within both clusters within the current deployment when fetching the configuration; use of the autodetect keyword will not collect the correct information.
   - **If you are upgrading any other deployment:**
     If you are using the $CONTINUENT_PROFILES variable to specify a location for your configuration, make sure that the variable has been set correctly.
     
     If you are not using $CONTINUENT_PROFILES, a copy of the existing configuration must be fetched from the installed Continuent Tungsten installation:
     ```shell
     ./tools/tpm fetch --hosts=host1,host2,host3,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 current configuration information will be retrieved to be used for the upgrade:
     ```shell
     ./tools/tpm fetch --hosts=host1,host2,host3 --user=tungsten --directory=/opt/continuent
     ```
     **NOTE** >> Configuration loaded from host1,host2,host3

5. Check that the update configuration matches what you expect by using tpm reverse:
   ```shell
   ./tools/tpm reverse
   # Options for the dsone data service
   tools/tpm configure dsone \
   --application-password=password \
   --application-port=3306 \
   --application-user=app_user \
   --connectors=host1,host2,host3 \
   --datasource-log-directory=/var/log/mysql \
   --install-directory=/opt/continuent \
   --master=host1 \
   --members=host1,host2,host3 \
   '--profile-script=~/.bashrc'
   ```
6. Run the upgrade process:

```bash
shell> ./tools/tpm update
```

**Note**

During the update process, **tpm** may report errors or warnings that were not previously reported as problems. This is due to new features or functionality in different MySQL releases and Continuent Tungsten updates. These issues should be addressed and the **tpm update** command re-executed.

The following additional options are available when updating:

- `--no-connectors` (optional)

By default, an update process will restart all services, including the connector. Adding this option prevents the connectors from being restarted. If this option is used, the connectors must be manually updated to the new version during a quieter period. This can be achieved by running on each host the command:

```bash
shell> tpm promote-connector
```

This will result in a short period of downtime (couple of seconds) only on the host concerned, while the other connectors in your configuration keep running. During the upgrade, the Connector is restarted using the updated software and/or configuration.

A successful update will report the cluster status as determined from each host in the cluster:

```
Getting cluster status on host1
Continuent Tungsten 2.0.5 build 3
connect to 'dsone@host1'
dsone: session established
[LOGICAL] /dsone > ls
COORDINATOR[host3:AUTOMATIC:ONLINE]

ROUTERS:
+----------------------------------------------------------------------------+
|connector@host1[31613](ONLINE, created=0, active=0)                      |
|connector@host2[27649](ONLINE, created=0, active=0)                      |
|connector@host3[21475](ONLINE, created=0, active=0)                      |
+----------------------------------------------------------------------------+
...```

The update process should now be complete. The current version can be confirmed by starting **cctrl**.

### 2.14.2. Upgrading when using INI-based configuration, or without ssh Access

To perform an upgrade of an individual node, **tpm** can be used on the individual host. The same method can be used to upgrade an entire cluster without requiring **tpm** to have **ssh** access to the other hosts in the dataservice.

To upgrade a cluster using this method:

1. Upgrade the slaves in the dataservice
2. Switch the current master to one of the upgraded slaves
3. Upgrade the master
4. Switch the master back to the original master

For more information on performing maintenance across a cluster, see Section 4.11.3, “Performing Maintenance on an Entire Dataservice”.

To upgrade a single host with `tpm`:
1. Download the release package.
2. Unpack the release package:
   ```bash
   shell> tar zxf continuent-tungsten-2.0.5-3.tar.gz
   ```
3. Change to the unpackaged directory:
   ```bash
   shell> cd continuent-tungsten-2.0.5-3
   ```
4. Execute `tpm update`, specifying the installation directory. This will update only this host:
   ```bash
   shell> ./tools/tpm update --directory=/opt/continuent
   ```

   **NOTE**
   >> Configuration loaded from host1
   -----------------------------
   Getting cluster status on host1
   Continuent Tungsten 2.0.5 build 3
   connect to 'dsone@host1'
   dsone: session established
   (LOGICAL) /dsone > ls
   
   #####################################################################
   # Next Steps
   #####################################################################
   We have added Tungsten environment variables to ~/.bashrc.
   Run `source ~/.bashrc` to rebuild your environment.

   Once your services start successfully you may begin to use the cluster.
   To look at services and perform administration, run the following command
   from any database server.

   `CONTINUENT_ROOT/tungsten/tungsten-manager/bin/cctrl`

   Configuration is now complete. For further information, please consult
   Tungsten documentation, which is available at docs.continuent.com.

   **NOTE**
   >> Command successfully completed

To update all of the nodes within a cluster, the steps above will need to be performed individually on each host.

### 2.14.3. Upgrading from Continuent Tungsten 1.5.3/1.5.4 to Continuent Tungsten 2.0

You can upgrade directly from Continuent Tungsten 1.5.3 or Continuent Tungsten 1.5.4 to Continuent Tungsten 2.0 using the standard update procedures.

When upgrading to Continuent Tungsten 2.0 from Continuent Tungsten 1.5.3/1.5.4, changes to the way witness hosts are used and identified mean that you should modify your configuration before completing the upgrade process.

The witness changes that affect the upgrade are as follows:
1. Witnesses must be on the same network subnet as the existing managers.
2. Dataservices must have at least three managers to provide status check during failure.
3. Active witnesses can be created; these install only the manager on target hosts to act witnesses to check network connectivity to the configured dataserver and connectors configured within the service.

If you are upgrading a host which satisfies conditions (1) and/or (2) in the above list, you can perform an upgrade using `tpm update`.

If your current witness host is not on the same network segment as the rest of your dataservice, the witness host must be removed from the configuration. Alternatively, you can add or update an existing witness host to be an active host witness within the cluster.

To upgrade a cluster and add the required active witnesses to the cluster, or to add additional active witnesses to a cluster which already has the required number of implied witness hosts, the configuration must be upgraded first before deploying the service.
Note

Active witnesses must have the prerequisites for hosts (Section C.2, "Host Configuration" configured before the update and deployment is completed.

1. On your staging server, download the release package.
2. Unpack the release package:
   
   shell> tar zxf continuent-tungsten-2.0.5-3.tar.gz

3. Change into the directory:
   
   shell> cd continuent-tungsten-2.0.5-3

4. Fetch a copy of the existing configuration information:
   
   shell> ./tools/tpm fetch --hosts=host1,host2,host3,autodetect --user=tungsten --directory=/opt/continuent

5. Update the configuration with additional witness hosts:
   
   shell> ./tools/tpm configure service_name --enable-active-witnesses=true --witnesses=hostname --members+=hostname

6. Run the update and installation process:
   
   shell> ./tools/tpm update service_name

If you have multiple services configured that require active witnesses, you must update each service with additional witness hosts.

2.14.4. Installing an Upgraded JAR Patch

Warning

The following instructions should only be used if Continuent Support have explicitly provided you with a customer JAR file designed to address a problem with your deployment.

If a custom JAR has been provided by Continuent Support, the following instructions can be used to install the JAR into your installation.

1. Determine your staging directory or untarred installation directory:
   
   shell> tpm query staging

Go to the appropriate host (if necessary) and the staging directory.

shell> cd continuent-tungsten-2.0.5-3

2. Change to the correct directory. For example, to update Tungsten Replicator change to tungsten-replicator/lib; for Tungsten Manager use tungsten-manager/lib; for Tungsten Connector use tungsten-connector/lib:
   
   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

2.15. Downgrading to an Earlier Release

If after upgrading Continuent Tungsten you are experiencing problems, and Continuent Support have suggested that you downgrade to an earlier version of Continuent Tungsten, follow these steps to revert your existing Continuent Tungsten installation.
1. Redirect all users directly to the MySQL server on the master. This may require changing applications and clients to point directly to the MySQL servers. You cannot use Tungsten Connector to handle this for you, since the entire cluster, including the Tungsten Connector services, will be removed.

2. Stop Tungsten services on all servers:

   ```shell
   stopall
   ```

3. Downgrading to Continuent Tungsten 2.0.x

   For Continuent Tungsten 2.0.x, the information stored in the database schema for the service, for example `tungsten_alpha`, can remain in place, unless you are changing the service name.

Downgrading to Continuent Tungsten 1.5.x

When downgrading to a release earlier than Continuent Tungsten 2.0, the schema used to hold information must be updated. You must rebuild the `tungsten` schema on all database servers in the updated cluster. This requires a number of different steps:

First, disable logging the statements to the binary log; this information does not need to be replicated around the cluster, even after restart:

```mysql
SET SESSION SQL_LOG_BIN=0;
```

Now delete the `tungsten` schema in preparation for it to be recreated. Within Continuent Tungsten 1.5.4, information about the replication state is stored in the `tungsten` schema; within Continuent Tungsten 2.0.1 the information is stored within a schema matching the service name, for example the service `alpha` would be stored in the schema `tungsten_alpha`.

```mysql
DROP SCHEMA IF EXISTS `tungsten`;
CREATE SCHEMA `tungsten`;
USE tungsten;
```

Now create the tables to store the status information:

```mysql
CREATE TABLE `consistency` (
    `db` char(64) NOT NULL DEFAULT '',
    `tbl` char(64) NOT NULL DEFAULT '',
    `id` int(11) NOT NULL DEFAULT '0',
    `row_offset` int(11) NOT NULL,
    `row_limit` int(11) NOT NULL,
    `this_crc` char(40) DEFAULT NULL,
    `this_cnt` int(11) DEFAULT NULL,
    `master_crc` char(40) DEFAULT NULL,
    `master_cnt` int(11) DEFAULT NULL,
    `ts` timestamp NULL DEFAULT NULL,
    `method` char(32) DEFAULT NULL,
    PRIMARY KEY (`db`,`tbl`,`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `heartbeat` (
    `id` bigint(20) NOT NULL DEFAULT '0',
    `seqno` bigint(20) DEFAULT NULL,
    `eventid` varchar(32) DEFAULT NULL,
    `source_tstamp` timestamp NULL DEFAULT NULL,
    `target_tstamp` timestamp NULL DEFAULT NULL,
    `lag_millis` bigint(20) DEFAULT NULL,
    `salt` bigint(20) DEFAULT NULL,
    `name` varchar(128) DEFAULT NULL,
    PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `history` (
    `seqno` bigint(20) NOT NULL DEFAULT '0',
    `fragno` smallint(6) NOT NULL DEFAULT '0',
    `last_frag` char(1) DEFAULT NULL,
    `source_id` varchar(128) DEFAULT NULL,
    `type` tinyint(4) DEFAULT NULL,
    `epoch_number` bigint(20) DEFAULT NULL,
    `source_tstamp` timestamp NULL DEFAULT NULL,
    `local_enqueue_tstamp` timestamp NULL DEFAULT NULL,
    `processed_tstamp` timestamp NULL DEFAULT NULL,
    `status` tinyint(4) DEFAULT NULL,
    `comments` varchar(128) DEFAULT NULL,
    `eventid` varchar(128) DEFAULT NULL,
    `event` longblob,
    PRIMARY KEY (`seqno`,`fragno`),
    KEY `eventid` (`eventid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `trep_commit_seqno` (
    `seqno` bigint(20) DEFAULT NULL,
    `ts` timestamp NULL DEFAULT NULL,
    `method` char(32) DEFAULT NULL,
    PRIMARY KEY (`seqno`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```
Now import the current sequence number from the existing Continuent Tungsten `trep_commit_seqno` table:

```sql
mysql> INSERT INTO tungsten.trep_commit_seqno
SELECT seqno, fragno, last_frag, source_id, epoch_number, eventid, applied_latency, update_timestamp
FROM TUNGSTEN_SERVICE_SCHEMA.trep_commit_seqno;
```

Check the sequence number:

```sql
mysql> SELECT * FROM tungsten.trep_commit_seqno;
```

If the sequence number doesn’t match on all servers, update the tungsten schema on the master with the earliest information (i.e. lowest sequence number):

```sql
mysql> SET SQL_LOG_BIN=0;
mysql> UPDATE tungsten.trep_commit_seqno SET seqno=###,epoch_number=###,eventid=SSSSS;
```

4. Extract the release of Continuent Tungsten that should be installed instead, and then using `tpm fetch` to retrieve the current configuration.

```shell
./tools/tpm fetch --user=tungsten --hosts=host1,host2,host3,host4
--release-directory=/opt/continuent
```

Note

In the event that the `tpm fetch` operation fails to detect the current configuration, run `tpm reverse` on one of the machines in the configured service. This will output the current configuration. If necessary, execute `tpm reverse` on multiple hosts to determine whether the information matches.

If you execute the returned text from `tpm reverse`, it will configure the service within the local directory, and the installation can then be updated.

Ensure that the current master is listed as the master within the configuration.

Now update Continuent Tungsten to deploy the new release:

```shell
./tools/tpm update
```

5. Start all the services on the master:

```shell
startall
```

Confirm that the current master is correct within `trepctl` and `ctrl`.

6. Start the services on remaining servers:

```shell
startall
```

7. If you were using a composite data service, you must recreate the composite dataservice configuration manually.

8. Once all the services are back up and running, it is safe to point users and applications at Tungsten Connector and return to normal operations.
Chapter 3. Advanced Deployments

3.1. Migrating and Seeding Data

3.1.1. Migrating from MySQL Native Replication 'In-Place'

If you are migrating an existing MySQL native replication deployment to use Continuent Tungsten the configuration of the Continuent Tungsten replication must be updated to match the status of the slave.

1. Deploy Continuent Tungsten using the model or system appropriate according to Chapter 2, Deployment. Ensure that the Continuent Tungsten is not started automatically by excluding the --start [248] or --start-and-report [248] options from the tpm commands.

2. On each slave

   Confirm that native replication is working on all slave nodes:

   ```shell
   echo 'SHOW SLAVE STATUS\G' | tpm mysql | \ 
   egrep ' Master_Host| Last_Error| Slave_SQL_Running'
   Master_Host: tr-ssl1
   Slave_SQL_Running: Yes
   Last_Error:
   ```

3. On the master and each slave

   Reset the Tungsten Replicator position on all servers:

   ```shell
   replicator start offline
   trepctl -service alpha reset -all -y
   ```

4. On the master

   Login and start Continuent Tungsten services and put the Tungsten Replicator online:

   ```shell
   startall
   trepctl online
   ```

5. On the master

   Put the cluster into maintenance mode using cctrl to prevent Continuent Tungsten automatically reconfiguring services:

   ```cctrl
   set policy maintenance
   ```

6. On each slave

   Record the current slave log position (as reported by the Master_Log_File and Exec_Master_Log_Pos output from SHOW SLAVE STATUS. Ideally, each slave should be stopped at the same position:

   ```shell
   echo 'SHOW SLAVE STATUS\G' | tpm mysql | \ 
   egrep ' Master_Host| Last_Error| Master_Log_File| Exec_Master_Log_Pos'
   Master_Host: tr-ssl1
   Master_Log_File: mysql-bin.000025
   Exec_Master_Log_Pos: 181268
   Last_Error: Error executing row event: 'Table 'tungsten_alpha.heartbeat' doesn't exist'
   ```

   If you have multiple slaves configured to read from this master, record the slave position individually for each host. Once you have the information for all the hosts, determine the earliest log file and log position across all the slaves, as this information will be needed when starting Continuent Tungsten replication. If one of the servers does not show an error, it may be replicating from an intermediate server. If so, you can proceed normally and assume this server stopped at the same position as the host is replicating from.

7. On the master

   Take the replicator offline and clear the THL:

   ```shell
   trepctl offline
   trepctl -service alpha reset -all -y
   ```

8. On the master

   Start replication, using the lowest binary log file and log position from the slave information determined in step 6.
9. On each slave
   a. Disable native replication to prevent native replication being accidentally started on the slave.

      *On MySQL 5.0 or MySQL 5.1:*

      ```shell>
      echo "STOP SLAVE; CHANGE MASTER TO MASTER_HOST='';" | tpm mysql
      ```

      *On MySQL 5.5 or later:*

      ```shell>
      echo "STOP SLAVE; RESET SLAVE ALL;" | tpm mysql
      ```

   b. If the final position of MySQL replication matches the lowest across all slaves, start Continuent Tungsten services:

      ```shell>
      trepgt online
      ```

      ```shell>
      startall
      ```

      The slave will start reading from the binary log position configured on the master.

      If the position on this slave is different, use `trepctl online -from-event` to set the online position according to the recorded position when native MySQL was disabled. Then start all remaining services with `startall`.

      ```shell>
      trepgt online -from-event 000025:188249
      ```

      ```shell>
      startall
      ```

10. Use `cctrl` to confirm that replication is operating correctly across the dataservice on all hosts.

11. Put the cluster back into automatic mode:

    ```cctrl>
    set policy automatic
    ```

12. Update your applications to use the installed connector services rather than a direct connection.

13. Remove the `master.info` file on each slave to ensure that when a slave restarts, it does not connect up to the master MySQL server again.

Once these steps have been completed, Continuent Tungsten should be operating as the replication service for your MySQL servers. Use the information in Chapter 4, *Operations Guide* to monitor and administer the service.

### 3.1.2. Migrating from MySQL Native Replication Using a New Service

When running an existing MySQL native replication service that needs to be migrated to a Continuent Tungsten service, one solution is to create the new Continuent Tungsten service, synchronize the content, and then install a service that migrates data from the existing native service to the new service while applications are reconfigured to use the new service. The two can then be executed in parallel until applications have been migrated.

The basic structure is shown in Figure 3.1, "Migration: Migrating Native Replication using a New Service". The migration consists of two steps:

- Initializing the new service with the current database state.
- Creating a Tungsten Replicator deployment that continues to replicate data from the native MySQL service to the new service.

Once the application has been switched and is executing against the new service, the secondary replication can be disabled by shutting down the Tungsten Replicator in `/opt/replicator`. 

---

## Advanced Deployments

Tungsten Replicator will start reading the MySQL binary log from this position, creating the corresponding THL event data.
Advanced Deployments

Figure 3.1. Migration: Migrating Native Replication using a New Service

To configure the service:
1.

Stop replication on a slave for the existing native replication installation :
mysql> STOP SLAVE;

Obtain the current slave position within the master binary log :
mysql> SHOW SLAVE STATUS\G
...
Master_Host: host3
Master_Log_File: mysql-bin.000002
Exec_Master_Log_Pos: 559
...

2.

Create a backup using any method that provides a consistent snapshot. The MySQL master may be used if you do not have a slave to
backup from. Be sure to get the binary log position as part of your back. This is included in the output to Xtrabackup or using the -master-data=2 option with mysqldump.

3.

Restart the slave using native replication :
mysql> START SLAVE;

4.

On the master and each slave within the new service, restore the backup data and start the database service

5.

Setup the new Continuent Tungsten deployment using the MySQL servers on which the data has been restored. For clarity, this will be
called newalpha.

6.

Configure a second replication service, beta to apply data using the existing MySQL native replication server as the master, and the
master of newalpha. The information provided in Section 2.8, “Replicating Data Into an Existing Dataservice” will help. Do not start the
new service.

7.

Set the replication position for beta using tungsten_set_position to set the position to the point within the binary logs where the
backup was taken:
shell> /opt/replicator/tungsten/tungsten-replicator/bin/tungsten_set_position \
--seqno=0 --epoch=0 --service=beta \
--source-id=host3 --event-id=mysql-bin.000002:559

8.

Start replicator service beta:
shell> /opt/replicator/tungsten/tungsten-replicator/bin/replicator start

Once replication has been started, use trepctl to check the status and ensure that replication is operating correctly.
The original native MySQL replication master can continue to be used for reading and writing from within your application, and changes
will be replicated into the new service on the new hardware. Once the applications have been updated to use the new service, the old
servers can be decommissioned and replicator service beta stopped and removed.

62


3.1.3. Seeding Data through MySQL

Once the Tungsten Replicator is installed, it can be used to provision all slaves with the master data. The slaves will need enough information in order for the installation to succeed and for Tungsten Replicator to start. The provisioning process requires dumping all data on the master and reloading it back into the master server. This will create a full set of THL entries for the slave replicators to apply. There may be no other applications accessing the master server while this process is running. Every table will be emptied out and repopulated so other applications would get an inconsistent view of the database. If the master is a MySQL slave, then the slave process may be stopped and started to prevent any changes without affecting other servers.

1. If you are using a MySQL slave as the master, stop the replication thread:

```
mysql> STOP SLAVE;
```

2. Check Tungsten Replicator status on all servers to make sure it is ONLINE and that the appliedLastSeqno values are matching:

```
shell> trepctl status
```

Starting the process before all servers are consistent could cause inconsistencies. If you are trying to completely reprovision the server then you may consider running `trepctl reset` before proceeding. That will reset the replication position and ignore any previous events on the master.

3. Use `mysqldump` to output all of the schemas that need to be provisioned:

```
shell> mysqldump --opt --skip-extended-insert -h host3 -u tungsten -P13306 -p --databases db1,db2 > ~/dump.sql
```

Optionally, you can just dump a set of tables to be provisioned:

```
shell> mysqldump --opt --skip-extended-insert -h host3 -u tungsten -P13306 -p db1 table1 table2 > ~/dump.sql
```

4. If you are using heterogeneous replication all tables on the slave must be empty before proceeding. The Tungsten Replicator does not replicate DDL statements such as `DROP TABLE` and `CREATE TABLE`. You may either truncate the tables on the slave or use `ddlscan` to recreate them.

5. Load the dump file back into the master to recreate all data:

```
shell> cat ~/dump.sql | tpm mysql
```

The Tungsten Replicator will read the binary log as the dump file is loaded into MySQL. The slaves will automatically apply these statements through normal replication.

6. If you are using a MySQL slave as the master, restart the replication thread after the dump file as completed loading:

```
mysql> START SLAVE;
```

7. Monitor replication status on the master and slaves:

```
shell> trepctl status
```

3.2. 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.

3.2.1. 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.

- 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 not show much improvement with parallel apply.

3.2.2. Enabling Parallel Apply

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`. The `--channels` option sets the number of channels (i.e., threads) you propose to use for applying data. Here is a code example of master-slave installation with parallel apply enabled. The slave will apply transactions using 30 channels.

```
shell> ./tools/tpm install --master-slave
     --master-host=logos1 \
     --datasource-user=tungsten \n     --datasource-password=secret \n     --service-name=myservice \n     --home-directory=/opt/continuent \n     --cluster-hosts=logos1,logos2 \n     --svc-parallelization-type=disk \n     --channels=30 \n     --start-and-report
```

There are several additional options that default to reasonable values. You may wish to change them in special cases.

• `--buffer-size` — Sets the replicator block commit size, which is the number of transactions to commit at once on slaves. Values up to 100 are normally fine.
• `--native-slave-takeover` — Used to allow Tungsten to take over from native MySQL replication and parallelize it. See here for more.

3.2.3. 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` 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.

3.2.4. 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 (disk option)**

A disk parallel queue uses a set of independent threads to read from the Transaction History Log and feed short in-memory queues used by channels. Disk queues have the advantage that they minimize memory required by Java. They also allow channels to operate some distance apart, which improves throughput. For instance, one channel may apply a transaction that committed 2 minutes before the transaction another channel is applying. This separation keeps a single slow transaction from blocking all channels.

Disk queues minimize memory consumption of the Java VM but to function efficiently they do require pages from the Operating System page cache. This is because the channels each independently read from the Transaction History Log. As long as the channels are close together the storage pages tend to be present in the Operating System page cache for all threads but the first, resulting in very fast reads. If channels become widely separated, for example due to a high `maxOfflineInterval` value, or the host has insufficient free memory, disk queues may operate slowly or impact other processes that require memory.

**Memory Parallel Queue (memory option)**

A memory parallel queue uses a set of in-memory queues to hold transactions. One stage reads from the Transaction History Log and distributes transactions across the queues. The channels each read from one of the queues. In-memory queues have the advantage that they do not need extra threads to operate, hence reduce the amount of CPU processing required by the replicator.

When you use in-memory queues you must set the `maxSize` property on the queue to a relatively large value. This value sets the total number of transaction fragments that may be in the parallel queue at any given time. If the queue hits this value, it does not accept further transaction fragments until existing fragments are processed. For best performance it is often necessary to use a relatively large number, for example 10,000 or greater.

The following example shows how to set the `maxSize` property after installation. This value can be changed at any time and does not require the replicator to go offline cleanly:

```bash
tpm update alpha
   --property=replicator.store.parallel-queue.maxSize=10000
```

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.

3.2.5. Parallel Replication and Offline Operation

3.2.5.1. 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.

3.2.5.2. 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.

```shell
./tools/tpm update alpha --property=replicator.store.parallel-queue.maxOfflineInterval:30
```

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.

### 3.2.5.3. 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.

### 3.2.6. Adjusting Parallel Replication After Installation

#### 3.2.6.1. 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:

```shell
./tools/tpm update alpha --channels=5
./replicator restart
```

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.

#### 3.2.6.2. 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:

```shell
./tpm update alpha --svc-parallelization-type=disk --channels=5
./replicator restart
```

### 3.2.7. Monitoring Parallel Replication

#### 3.2.7.1. 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>trepctl status</td>
<td>Shows basic variables including overall latency of slave and number of apply channels</td>
</tr>
</tbody>
</table>
### 3.2.7.2. 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.

```shell
shell> trepctl status
Processing status command...
NAME VALUE
---- -----
appliedLastEventId : mysql-bin.000211:0000000020094456;0
appliedLastSeqno : 78021
appliedLatency : 0.216
channels : 5
...
Finished status command...
```

When parallel apply is enabled, the meaning of `appliedLastSeqno` changes. It is the minimum recovery position across apply channels, which means it is the position where channels restart in the event of a failure. This number is quite conservative and may make replication appear to be further behind than it actually is.

- Busy channels mark their position in table `trep_commit_seqno` as they commit. These are up-to-date with the traffic on that channel, but channels have latency between those that have a lot of big transactions and those that are more lightly loaded.

- Inactive channels do not get any transactions, hence do not mark their position. Tungsten sends a control event across all channels so that they mark their commit position in `trep_commit_channel`. It is possible to see a delay of many seconds or even minutes in unloaded systems from the true state of the 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:

```shell
shell> tpm update alpha --property=replicator.store.parallel-queue.syncInterval=500
```

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
shell> trepctl status -name stores
Processing status command (stores)...
NAME VALUE
---- -----
name : parallel-queue
...
storeClass : com.continuent.tungsten.replicator.thl.THLParallelQueue
syncInterval : 10000
Finished status command (stores)...
```

You can also force all channels to mark their current position by sending a heartbeat through using the `trepctl heartbeat` command.

### 3.2.7.3. Relative Latency

Relative latency is a `trepctl status` parameter. It indicates the latency since the last time the appliedSeqno advanced; for example:

```shell
shell> trepctl status
Processing status command...
NAME VALUE
---- -----
appliedLastEventId : mysql-bin.000211:0000000020094766;0
appliedLastSeqno : 78022
appliedLatency : 0.571
relativeLatency : 8.944
```
3.2.7.4. Serialization Count

Serialization count refers to the number of transactions that the replicator has handled that cannot be applied in parallel because they involve dependencies across shards. For example, a transaction that spans multiple shards must serialize because it might cause cause an out-of-order update with respect to transactions that update a single shard only.

You can detect the number of transactions that have been serialized by looking at the `serializationCount` parameter using the `trepctl status -name stores` command. The following example shows a replicator that has processed 1512 transactions with 26 serialized.

```bash
shell> trepctl status -name stores
Processing status command (stores)...
... NAME  VALUE ----  ----- criticalPartition : -1 discardCount : 0 estimatedOfflineInterval: 0.0 eventCount: 1512 headSeqno: 78022 maxOfflineInterval: 5 maxsize: 10 name: parallel-queue queues: 5 serializationCount: 26 serialized: false ...
Finished status command (stores)...
```

In this case 1.7% of transactions are serialized. Generally speaking you will lose benefits of parallel apply if more than 1-2% of transactions are serialized.

3.2.7.5. 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.

```bash
shell> tpm update alpha --property=replicator.store.parallel-queue.maxOfflineInterval=15
```

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:

```bash
shell> trepctl status -name stores
Processing status command (stores)...
... NAME  VALUE ----  ----- estimatedOfflineInterval: 1.3 ...
maxOfflineInterval  1.5 ...
Finished status command (stores)...
```

3.2.7.6. 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.

```bash
shell> trepctl status -name shards
Processing status command (shards)...
... NAME  VALUE ----  ----- appliedLastEventId: mysql-bin.000211:0000000200950760 appliedLastSeqno: 78023 appliedLatency: 0.255 eventCount: 13923
```
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 `lastAppliedSeqno` for this shard gives the sequence number of the most recent serialized transaction. As the following example shows, you can then list the contents of the transaction to see why it serialized. In this case, the transaction affected tables in different schemas.

```
shell> trepctl status -name shards
Processing status command (shards)...
NAME   VALUE
----   -----  
appliedLastEventId: mysql-bin.000211:000000020095529;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.000211:000000020095529;0
- SOURCEID = logos1
- METADATA = [mysql_server_id=1;service=percona;shard=#UNKNOWN]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [**##charset = ISO8859_1, autocommit = 1, sql_auto_is_null = 0,»
  foreign_key_checks = 1, unique_checks = 1, sql_mode = '', character_set_client = 8, »
  collation_connection = 8, collation_server = 33]
- SCHEMA = 
- SQL(0) = insert into mats_0.foo values(1) /* ___SERVICE___ = [percona] */
- OPTIONS = [**##charset = ISO8859_1, autocommit = 1, sql_auto_is_null = 0,»
  foreign_key_checks = 1, unique_checks = 1, sql_mode = '', character_set_client = 8, »
  collation_connection = 8, collation_server = 33]
- SQL(1) = insert into mats_1.foo values(1)
```

The replicator normally distributes shards evenly across channels. As each new shard appears, it is assigned to the next channel number, which then rotates back to 0 once the maximum number has been assigned. If the shards have uneven transaction distributions, this may lead to an uneven number of transactions on the channels. To check, use the `trepctl status -name tasks` and look for tasks belonging to the `q-to-dbms` stage.

```
shell> trepctl status -name tasks
Processing status command (tasks)...
NAME   VALUE
----   -----  
appliedLastEventId: mysql-bin.000211:000000020095076;0
appliedLastSeqno : 78023
appliedLatency   : 0.248
applyTime       : 0.003
averageBlockSize: 2.520
cancelled       : false
currentLastEventId: mysql-bin.000211:000000020095076;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.

### 3.2.8. 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 `shared=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.

### 3.3. Deploying SSL Secured Replication and Administration

Continuent Tungsten supports encrypted communication between replication hosts. SSL can be employed at two different levels within the configuration, encryption of the THL communication channel used to transfer database events, and encryption (and implied authentication) of the JMX remote method invocation (RMI) used to administer services remotely within Continuent Tungsten.

To use SSL you must be using a Java Runtime Environment or Java Development Kit 1.5 or later. SSL is implemented through the `javax.net.ssl.SSLServerSocketFactory` socket interface class.

You will also need an SSL certificate. These can either be self-generated or obtained from an official signing authority. The certificates themselves must be stored within a Java keystore and truststore. To create your certificates and add them to the keystore or truststore, see Section 3.3.1, "Creating the Truststore and Keystore". Instructions are provided for self-generated, self-signed, and officially signed versions of the necessary certificates.

For JMX RMI authentication, a password file and authentication definition must also be generated. This information is required by the JMX system to support the authentication and encryption process. See Section 3.3.2, "SSL and Administration Authentication" for more information.

Once the necessary files are available, you need to use `tpm` to install, or update an existing installation with the SSL configuration. See Section 3.3.3, "Configuring the Secure Service through `tpm`".
Although not strictly required for installation, it may be useful to have the OpenSSL package installed. This contains a number of tools and utilities for dealing with certificate authority and general SSL certificates.

### 3.3.1. 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 Continuent Tungsten 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 Continuent Tungsten, 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.

#### 3.3.1.1. Creating Your Own Client and Server Certificates

Because the client and server components of the Continuent Tungsten 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 Continuent Tungsten 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 Continuent Tungsten configuration.

```
shell> keytool -import -v -trustcacerts -alias replserver -file client.cer -keystore truststore.ts

Enter keystore password:
Re-enter new password:
Owner: CN=My Name, OU=My OU, O=Continuent, L=Mountain View, ST=CA, C=US
Issuer: CN=My Name, OU=My OU, O=Continuent, L=Mountain View, ST=CA, C=US
Serial number: 87db1e1
Valid from: Wed Jul 31 17:15:05 BST 2013 until: Tue Oct 29 16:15:05 GMT 2013
Certificate fingerprints:
Signature algorithm name: SHA256withRSA
Version: 3

Extensions:
  #1: ObjectId: 2.5.29.14 Criticality=false
  SubjectKeyIdentifier [K]:
    0000: E7 51 0B 42 AC 61 84   D4 2E 9A F1 88 00 88 44 ....B.a........D
    0010: E4 69 C6 C7                                        .i..
  
Trust this certificate? [no]: yes
Certificate was added to keystore
[Storing truststore.ts]
```

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 `tpm` to configure the cluster. See Section 3.3.3, “Configuring the Secure Service through `tpm`”.

### 3.3.1.2. 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? [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
```

A non-interactive version is available by using the `-noprompt` option and supplying the truststore name:

```
shell> keytool -genkey -alias replserver -keyalg RSA -keystore keystore.jks
```

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 Continuent Tungsten configuration.

```
shell> keytool -import -v -trustcacerts -alias replserver -file client.cer -keystore truststore.ts

Enter keystore password:
Re-enter new password:
Owner: CN=My Name, OU=My OU, O=Continuent, L=Mountain View, ST=CA, C=US
Issuer: CN=My Name, OU=My OU, O=Continuent, L=Mountain View, ST=CA, C=US
Serial number: 87db1e1
Valid from: Wed Jul 31 17:15:05 BST 2013 until: Tue Oct 29 16:15:05 GMT 2013
Certificate fingerprints:
Signature algorithm name: SHA256withRSA
Version: 3

Extensions:
  #1: ObjectId: 2.5.29.14 Criticality=false
  SubjectKeyIdentifier [K]:
    0000: E7 51 0B 42 AC 61 84   D4 2E 9A F1 88 00 88 44 ....B.a........D
    0010: E4 69 C6 C7                                        .i..
  
Trust this certificate? [no]: yes
Certificate was added to keystore
[Storing truststore.ts]
```

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 `tpm` to configure the cluster. See Section 3.3.3, “Configuring the Secure Service through `tpm`”.
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): 

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 3.3.1.4, “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:
Advanced Deployments

```
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. See Section 3.3.3, "Configuring the Secure Service through tpm".

### 3.3.1.3. 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 3.3.1.4, "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:

```
shell> keytool -import -trustcacerts -alias replserver -file client.cer -keystore truststore.ts -storepass password -noprompt
shell> keytool -import -trustcacerts -alias replserver -file cacert.pem -keystore truststore.ts -storepass password -noprompt
shell> keytool -import -trustcacerts -alias replserver -file intercert.pem -keystore truststore.ts -storepass password -noprompt
```

### 3.3.1.4. 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 Continuent Tungsten.
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.

### 3.3.2. SSL and Administration Authentication

Continuent Tungsten uses JMX RMI to perform remote administration and obtain information from remote hosts within the dataservice. This communication can be encrypted and authenticated.

To configure this operation two files are required, one defines the authentication configuration, the other configures the username/password combinations used to authenticate. These files and configuration are used internally by the system to authenticate.

The authentication configuration defines the users and roles. The file should match the following:

```
monitorRole   readonly
controlRole   readwrite \ 
create javax.management.monitor.*,javax.management.timer.* \ 
unregister

unregister
```

The contents or description of this file must not be changed. Create a file containing this information in your configuration, for example `jmxremote.access`

Now a corresponding password configuration must be created using the `tpasswd` tool (located in `cluster-home/bin/tpasswd`). By default, plain-text passwords are generated:

```
shell> tpasswd -c tungsten password
-t rmi_jmx
-p password.store
-tsp truststore.ts
```

To use encrypted passwords, the truststore and truststore password must be supplied so that the certificate can be loaded and used to encrypt the supplied password. The `-e` must be specified to encrypt the password:

```
shell> tpasswd -c tungsten password
-t rmi_jmx
-p password.store
-e
-tsp truststore.ts
```

This creates a user, `tungsten`, with the password `password` in the file `password.store`.

The password file, and the JMX security properties file will be needed during configuration. See Section 3.3.3, "Configuring the Secure Service through `tpm`".
3.3.3. Configuring the Secure Service through tpm

To configure a basic SSL setup where the THL communication between, the keystore, truststore, and corresponding passwords must be configured in your installation.

**Configuring SSL for THL Only**

The configuration can be applied using `tpm`, either during the initial installation, or when performing an update of an existing installation. The same command-line options should be used for both. For the keystore and truststore, the pathnames supplied to `tpm` will be distributed to the other hosts during the update.

For example, to update an existing configuration, go to the staging directory for your installation:

```
shell> ./tools/tpm update \
    --thl-ssl=true \ 
    --java-keystore-path=~/keystore.jks \ 
    --java-keystore-password=password \ 
    --java-truststore-path=~/truststore.ts \ 
    --java-truststore-password=password
```

Where:

- `--thl-ssl`  
  This enables SSL encryption on for THL when set to `true`.

- `--java-keystore-path`  
  Sets the location of the certificate keystore, the file will be copied to the installation directory during configuration.

- `--java-keystore-password`  
  The password for the keystore.

- `--java-truststore-path`  
  Sets the location of the certificate truststore, the file will be copied to the installation directory during configuration.

- `--java-truststore-password`  
  The password for the truststore.

**Note**

If you plan to update your configuration to use RMI authentication with SSL, the keystore and truststore must be the same as that used for THL SSL.

Once the installation or update has completed, the use of SSL can be confirmed by checking the THL URIs used to exchange information. For secure communication, the protocol is `thls`, as in the example output from `trepctl status`:

```
shell> trepctl status
    Processing status command...
    NAME                     VALUE
    ----                     -----  
    appliedLastEventId     : mysql-bin.000011:0000000000003097;0
    ... 
    masterConnectUri       : thls://localhost/ 
    masterListenUri        : thls://tr-ms1:2112/ 
    maximumStoredSeqNo     : 15 
    minimumStoredSeqNo     : 0
    ... 
    Finished status command...
```

**Configuring SSL for Administration**

Authentication and SSL encryption for administration controls the communication between administration tools such as `cctrl`. This prevents unknown tools for attempting to use the JMX remote invocation to perform different administration tasks.

The system works by encrypting communication, and then using explicit authentication (defined by the RMI user) to exchange authentication information.

To update your existing installation, go to the staging directory for your installation:

```
shell> ./tools/tpm update \
```
Where:

- **--rmi-ssl** [232]
  
  If set to `true`, enables RMI SSL encryption.

- **--rmi-authentication** [232]
  
  If set to `true`, enables authentication for the RMI service.

- **--rmi-user** [246]
  
  The user that will be used when performing administration. This should match the username used when creating the password file and security properties.

- **--java-jmxremote-access-path** [235]
  
  The path to the file containing the JMX RMI configuration, as configured in Section 3.3.2, “SSL and Administration Authentication”.

- **--java-passwordstore-path** [236]
  
  The location of the password file created when setting the password, as described in Section 3.3.2, “SSL and Administration Authentication”.

- **--java-keystore-path** [236]
  
  Sets the location of the certificate keystore, the file will be copied to the installation directory during configuration.

- **--java-keystore-password** [235]
  
  The password for the keystore.

- **--java-truststore-path** [236]
  
  Sets the location of the certificate truststore, the file will be copied to the installation directory during configuration.

- **--java-truststore-password** [236]
  
  The password for the truststore.

Once the update or installation has been completed, check that `trepctl` works and shows the status.

### SSL Settings During an Upgrade

When updating an existing installation to a new version of Continuent Tungsten, the installation uses the existing configuration parameters for SSL and authentication. If the original files from their original locations still exist they are re-copied into the new installation and configuration. If the original files are unavailable, the files from the existing installation are copied into the new installation and configuration.

### Configuring SSL for THL and Administration

To configure both JMX and THL SSL encrypted communication, you must specify the SSL and JMX security properties. The SSL properties are the same as those used for enabling SSL on THL, but adding the necessary configuration parameters for the JMX settings:
This configures SSL and security for authentication. These options for `tpm` can be used to update an existing installation, or defined when creating a new deployment.

**Important**

All SSL certificates have a limited life, specified in days when the certificate is created. In the event that your replication service fails to connect, check your certificate files and confirm that they are still valid. If they are out of date, new certificates must be created, or your existing certificates can be renewed. The new certificates must then be imported into the keystore and truststore, and `tpm update` executed to update your replicator configuration.

### 3.3.4. Configuring Connector SSL

SSL communication is supported for Tungsten Connector in three different possible combinations:

- SSL from the application to Tungsten Connector; Non-SSL connections from Tungsten Connector to MySQL
- Non-SSL from the application to Tungsten Connector; SSL connections from Tungsten Connector to MySQL
- SSL from the application to Tungsten Connector; SSL connections from Tungsten Connector to MySQL

The connector also supports application connections using either SSL or Non-SSL communication on the same TCP/IP port. This allows you to choose SSL communication without changing your application ports.

To enable SSL communication with Tungsten Connector you must create suitable certificates keys and keystores, as described in Section 3.3.1, “Creating the Truststore and Keystore”. The keystores used for Tungsten Connector can be the same, or different, to the keystores used for securing the manager and replication communication.

To enable connector SSL during installation or update, the `--connector-ssl=true` option must be set to true:

```shell
../tools/tpm update service_name --connector-ssl=true \
--java-connector-keystore-path=/home/tungsten/keystore.jks \
--java-connector-keystore-password=password \
--java-connector-truststore-path=/home/tungsten/truststore.ts \
--java-connector-truststore-password=password
```

This will update the connector configuration with the specified keystores, truststore and enable SSL on the connector connections.

### 3.3.5. Connector SSL Example Procedure

The below procedures accomplish the following objectives:

- Create, activate and test SSL keys for the MySQL server
- Enable and test SSL encrypted traffic between the MySQL server and the Connector
- Enable and test SSL encrypted traffic between the Application/Client and the Connector

#### 3.3.5.1. Setup Environment and Paths

Use these environment values on all Database & Connector nodes.

```shell
shell> export CONN_CERTS_PATH=/opt/continuent/share
shell> export MYSQL_CONFIG_PATH=/etc/mysql
shell> export MYSQL_CERTS_PATH=$MYSQL_CONFIG_PATH/certs
```

The `certs` directory is required on all Database nodes to hold the MySQL server certificates and keys.

```shell
shell> sudo mkdir -p $MYSQL_CERTS_PATH
shell> sudo chown mysql: $MYSQL_CERTS_PATH
shell> sudo chmod 775 $MYSQL_CERTS_PATH
```

#### 3.3.5.2. Configuring SSL for MySQL Server

**Important**

The "Common Name" field for the Server and Client certificates MUST be different than the "Common Name" specified for the CA Cert.

1. Generate CA Cert
Advanced Deployments

2. Generate Server Cert

```bash
shell> openssl genrsa 2048 > $MYSQL_CERTS_PATH/ca-key.pem
shell> openssl req -new -x509 -nodes -days 3600
   -key $MYSQL_CERTS_PATH/ca-key.pem
   -out $MYSQL_CERTS_PATH/ca-cert.pem
```

3. Generate Client Cert

```bash
shell> openssl req -newkey rsa:2048 -days 3600 -nodes
   -keyout $MYSQL_CERTS_PATH/server-key.pem
   -out $MYSQL_CERTS_PATH/server-req.pem
shell> openssl rsa -in $MYSQL_CERTS_PATH/server-key.pem -out $MYSQL_CERTS_PATH/server-key.pem
shell> openssl x509 -req -in $MYSQL_CERTS_PATH/server-req.pem -days 3600
   -CA $MYSQL_CERTS_PATH/ca-cert.pem
   -CAkey $MYSQL_CERTS_PATH/ca-key.pem
   -set_serial 01
   -out $MYSQL_CERTS_PATH/server-cert.pem
```

4. Verify All Certs

```bash
shell> openssl verify -CAfile $MYSQL_CERTS_PATH/ca-cert.pem
   $MYSQL_CERTS_PATH/server-cert.pem $MYSQL_CERTS_PATH/client-cert.pem
```

5. Copy certs to all Database nodes (repeat as needed so that every Database node has the same certs)

```bash
shell> rsync -av $MYSQL_CONFIG_PATH/ yourDBhost:$MYSQL_CONFIG_PATH/
```

6. Set proper ownership and permissions on ALL DB nodes

```bash
shell> sudo chown -R mysql: $MYSQL_CONFIG_PATH/
shell> sudo chmod -R g+w $MYSQL_CONFIG_PATH/
```

7. Update the my.cnf file to include the SSL certs you just created (add three lines to the [mysqld] stanza)

```bash
shell> vi /etc/my.cnf
[mysqld]
...
port=13306
# add three lines for SSL support
ssl-ca=/etc/mysql/certs/ca-cert.pem
ssl-cert=/etc/mysql/certs/server-cert.pem
ssl-key=/etc/mysql/certs/server-key.pem
...
```

8. Restart MySQL on all nodes using the standard rolling maintenance procedure - see Section 4.11.3, “Performing Maintenance on an Entire Database” for more information.

```bash
cctrl> ls
cctrl> datasource db3 shun
db3# service mysql restart
cctrl> recover
cctrl> datasource db2 shun
db2# service mysql restart
cctrl> recover
cctrl> switch to db2
cctrl> datasource db1 shun
db1# service mysql restart
cctrl> recover
cctrl> switch to db1
cctrl> ls
```
9. Add a new user to MySQL that requires SSL to connect. Do this just once on the current Master and let it propagate to the slaves.

```sql
shell> tpm mysql
mysql> DROP USER ssl_user;
mysql> CREATE USER ssl_user@'%' IDENTIFIED BY 'secret';
mysql> GRANT ALL PRIVILEGES ON *.* TO ssl_user@'%' REQUIRE SSL WITH GRANT OPTION;
mysql> flush privileges;
```

10. Verify that MySQL is working with SSL

a. Expect this to fail, because the ssl_user is only allowed to connect to the database using SSL:

```shell
shell> mysql -u ssl_user -psecret -h 127.0.0.1 -P 13306
```

b. Expect this to pass, because we have supplied the proper SSL credentials:

```shell
shell> mysql -u ssl_user -psecret -h 127.0.0.1 -P 13306 --ssl-ca=/etc/mysql/certs/ca-cert.pem
```

c. Verify SSL:

```sql
mysql> status
... SSL: Cipher in use is DHE-RSA-AES256-SHA
... 
```

**Important**

If you are able to login to MySQL and see that the status is SSL: Cipher in use, then you have successfully configured MySQL to use SSL.

### 3.3.5.3. Enable and Test SSL encryption from the Connector to the Database

1. Convert MySQL Client Cert to pkcs12 format

```shell
shell> openssl pkcs12 -export \
    -inkey $MYSQL_CERTS_PATH/client-key.pem \
    -in $MYSQL_CERTS_PATH/client-cert.pem \
    -out $MYSQL_CERTS_PATH/client-cert.p12 \
    -passout pass:secret
```

2. Create `tungsten_connector_keystore.jks`

```shell
shell> keytool -importkeystore \
    -srckeystore $MYSQL_CERTS_PATH/client-cert.p12 \
    -srcstoretype PKCS12 \
    -destkeystore $CONN_CERTS_PATH/tungsten_connector_keystore.jks \
    -deststorepass secret \
    -srcstorepass secret
```

3. Import the CA Cert into the KeyStore

```shell
shell> keytool -import -alias mysqlServerCACert -file $MYSQL_CERTS_PATH/ca-cert.pem \
    -keystore $CONN_CERTS_PATH/tungsten_connector_keystore.jks \
    -storepass secret -noprompt
```

4. Import the CA Cert into the TrustStore

```shell
shell> keytool -import -alias mysqlServerCACert -file $MYSQL_CERTS_PATH/ca-cert.pem \
    -keystore $CONN_CERTS_PATH/tungsten_connector_truststore.ts \
    -storepass secret -noprompt
```

5. Copy certs to all Connector nodes (repeat as needed so that every Connector node has the same certs)

```shell
shell> rsync -av $CONN_CERTS_PATH/ yourDBhost:$MYSQL_CONFIG_PATH/
```

6. Set proper ownership and permissions on ALL Connector nodes

```shell
shell> sudo chown tungsten: $CONN_CERTS_PATH/
```

7. Add the new MySQL user to the Connector’s `user.map` config file.

See Section 5.4.1, "user.map File Format" for more information.

```shell
shell> vi /opt/continuent/tungsten/tungsten-connector/conf/user.map
ssl_user secret theSvcName
```

8. Update the Connector configuration to enable SSL
• **Staging Method**

Update all nodes (DB & Connector) in the cluster

```
shell> tpm query staging
shell> cd (STAGING_DIR)
shell> tools/tpm configure (yourServiceName) \
    --connector-ssl=true \
    --java-connector-keystore-password=secret \
    --java-connector-truststore-password=secret \
    --java-connector-truststore-path=/opt/continuent/share/tungsten_connector_truststore.ts \
    --java-connector-keystore-path=/opt/continuent/share/tungsten_connector_keystore.jks
shell> tools/tpm update
```

• **INI Method**

Repeat these two steps on each node (DB & Connector)

```
shell> vi /etc/tungsten/tungsten.ini
[defaults]
...
# enable SSL from the connector to the DB
connector-ssl=true
java-connector-keystore-password=secret
java-connector-truststore-password=secret
java-connector-truststore-path=/opt/continuent/share/tungsten_connector_truststore.ts
java-connector-keystore-path=/opt/continuent/share/tungsten_connector_keystore.jks
...
shell> tpm update
```

9. **Test SSL connectivity through the connector**

a. Connect as the default application user

```
shell> tpm connector
```

b. Check the connection status

**Note**

**Expecting "SSL.IN=false SSL.OUT=true"**

SSL.IN is false because the the tpm connector command calls the mysql client in non-SSL mode.

SSL.OUT is true because the connection to the database is encrypted, even if the connection from the mysql client is not.

This can be verified with the "sudo tcpdump -X port 13306" command. Without the encryption, queries and responses are sent in plaintext and are visible in the output of tcpdump. When encryption is enabled, the queries and results are no longer visible.

```
mysql> tungsten connection status;
+-----------------------------------------------------------------------------+
| Message                                                                     |
+-----------------------------------------------------------------------------+
| db1@east(master:ONLINE) STATUS(OK), QOS=RW STRICT SSL.IN=false SSL.OUT=true |
+-----------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

c. Check the SSL status

**Note**

**Expecting "SSL: Not in use"**

SSL is not in use because the the tpm connector command calls the mysql client in non-SSL mode.

The connection to the database is encrypted, even if the connection from the mysql client is not.

This can be verified with the "sudo tcpdump -X port 13306" command. Without the encryption, queries and responses are sent in plaintext and are visible in the output of tcpdump. When encryption is enabled, the queries and results are no longer visible.

```
mysql> status
```
Important
If you are able to login to MySQL and see that the "tungsten connection status;" is SSL.OUT=true, then you have successfully configured the communication between the Connector and MySQL to use SSL.

3.3.5.4. Test SSL encryption from the Application to the Database

1. Connect as the SSL-enabled application user through the Connector host

```shell
mysql -u ssl_user -psecret -h 127.0.0.1 -P 3306 --ssl-ca=/etc/mysql/certs/ca-cert.pem
```

2. Check the connection status

   **Note**

   **Expecting "SSL.IN=true SSL.OUT=true"

   SSL.IN is true because the mysql client was invoked in SSL mode. Communications from the mysql client to the connector are encrypted.

   SSL.out is true because the connection to the Database from the Connector is encrypted.

```sql
mysql> tungsten connection status;
```

3. Check the SSL status

   **Note**

   **Expecting "Cipher in use is XXX-XXXX-XXXXXX-XXX"

   SSL is in use because the mysql client was invoked in SSL mode.

   The connection from the mysql client to the database is encrypted.

```sql
mysql> status
```
Important

If you are able to login to MySQL and see that the "tungsten connection status;" is "SSL.IN=true SSL.OUT=true", and the "status;" contains "Cipher in use is XXX-XXX-XXXXXX-XXX", then you have successfully configured SSL-encrypted communication between the Application/Client and MySQL through the Connector.
Chapter 4. Operations Guide

Continuent Tungsten™ has a wide range of tools and functionality available for checking and managing the status of a dataservice. The majority of the management and information structure is based around a small number of command-line utilities that provide a complete range of tools and information, either through a direct command-line, or secondary shell-like interface.

The main tool for controlling dataservices is cctrl. This provides a shell-like interface for querying and managing the dataservice and includes shell-like features such as command history and editing. Commands can be executed using cctrl either interactively:

```
shell> cctrl
connect to 'alpha@host1'
alpha: session established
[LOGICAL:EXPERT] /alpha > ls
```

Or by supplying a command and piping that as input to the cctrl shell:

```
shell> echo 'ls' | cctrl
```

The cctrl command is designed to provide information and management of the dataservice. Warnings and confirmations will be provided if a particular operation is potentially dangerous to the normal operation of the dataservice. These warnings can be disabled by switch to expert mode, either on the command-line:

```
shell> cctrl -expert
```

Or by changing the mode within cctrl:

```
[LOGICAL:EXPERT] /alpha > expert
WARNING: This is an expert-level command:
Incorrect use may cause data corruption
or make the dataservice unavailable.
Do you want to continue? [y/n] > y
[LOGICAL:EXPERT] /alpha >
```

When in expert mode, the mode is shown within the prompt.

When installing the dataservice using tpm, if requested, the login script for the staging user (for example .bashrc) will have been updated to execute a script within the installation directory called env.sh. This configures the location of the installation, configuration, and adds the script and binary directories to the PATH so that the commands can be executed without having to use the full path to the tools.

If the script was not added to the login script automatically, or needs to be added to the current session, the script is located within the share directory of the installation directory. For example, /opt/continuent/share/env.sh. To load into the current session use source.

```
shell> source /opt/continuent/share/env.sh
```

4.1. Checking Dataservice Status

The cctrl command provides the main interface to the dataservice information and control. The current status and configuration of the dataservice can be determined by using the ls command within the cctrl shell:

```
shell> cctrl
Continuent Tungsten 2.0.1 build 161
connect to 'alpha@host1'
alpha: session established
[LOGICAL:EXPERT] /alpha > ls
COORDINATOR[host1:AUTOMATIC:ONLINE]
ROUTERS:
| connector@host1[8805](ONLINE, created=0, active=0) |
| connector@host2[12039](ONLINE, created=0, active=0) |
| connector@host3[12712](ONLINE, created=0, active=0) |
DATASOURCES:
| host1(master:ONLINE, progress=3, THL latency=0.561) |
| STATUS [OK] [2013/05/03 09:11:10 PM BST] |
| MANAGER(state=ONLINE) |
| REPLICATOR(role=master, state=ONLINE) |
| DATASERVER(state=ONLINE) |
| CONNECTIONS(created=0, active=0) |
```
The output consists of the following major sections:

- **COORDINATOR**

  The coordinator is the node in the dataservice that is acting as the manager for the dataservice. The coordinator is decided upon within the dataservice by a consensus agreement, and the coordinator can change in the event of a failure of the existing coordinator. The coordinator is always the oldest datasource within the group that manages the dataservice, and does not need to be the same host as the master.

  The information about the coordinator is described in the following square brackets as `HOSTNAME:POLICY:STATUS`, where:

  - **HOSTNAME**
    - The hostname of the current coordinator.

  - **POLICY**
    - The current policy manager mode, which describes how the manager will respond to certain events. For example, in `AUTOMATIC` mode the manager will respond to issues and problems automatically, for example by performing an automatic master switch during a failover event.

    For more information on policy modes, see Section 4.2, “Policy Modes”.

  - **STATUS**
    - The current status of the coordinator host.

- **ROUTERS**

  A list of the currently configured SQL routers (using Tungsten Connector™) that are directing queries to the datasources. In the example, the dataservice consists of three routers, each connected to all of the configured data sources. The information output includes a summary of the number of connections made through the router, and the number of active connections to each router.

- **DATASOURCES**

  The `DATASOURCES` section lists detailed information providing one block for each configured datasource. The header block of the datasource output describes the overall status of the datasource:

  ```
  [host2(slave:ONLINE, progress=3, latency=1.243)]
  [STATUS [OK] [2013/05/04 05:40:43 AM BST]]
  ```

  - **MANAGER(state=ONLINE)**
  - **REPLICATOR(role=slave, master=host1, state=ONLINE)**
  - **DATASERVER(state=ONLINE)**
  - **CONNECTIONS(created=0, active=0)**

  ```
  [host3(slave:ONLINE, progress=3, latency=0.000)]
  [STATUS [OK] [2013/05/04 07:40:12 AM BST]]
  ```

  - **MANAGER(state=ONLINE)**
  - **REPLICATOR(role=slave, master=host1, state=ONLINE)**
  - **DATASERVER(state=ONLINE)**
  - **CONNECTIONS(created=0, active=0)**

  The first line describes the host and status information:

  - **Hostname** of the datasource (`host1`)

  - **Current role** within the dataservice and status of the datasource. For more information on roles, see Section 4.1.3, “Understanding Datasource Roles”. For information on datasource states, see Section 4.1.4, “Understanding Datasource States”.

  - **The `progress`** indicates the current sequence number from the THL for the datasource.

  - **The `THL latency`** Shows the current latency of the datasource. For a master datasource using MySQL, this is the latency between the data being written to the MySQL binary log and being processed in the THL. For a slave, it shows the latency between the original commit (from the master) and the application on the slave.
The second line provides a more detailed current status, and the time since the status was last changed. In the event of a change of status, for example to the SHUNNED or OFFLINE state, the time will indicate how long the node has been in that status.

- The remaining lines of the datasource description provide detailed information about each of the remaining services on the datasource and their status. The list will depend on the assigned roles and parameters for each datasource. It is important to note that each service has a status that is independent of the overall datasource status.

<table>
<thead>
<tr>
<th>MANAGER(state=ONLINE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Manager service, and the current status of the manager. If a configured datasource is down, has recently been restarted, or the manager has been stopped, the status may be offline.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>REPLICATOR(role=slave, master=host1, state=ONLINE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Tungsten Replicator service, which replicates data between hosts. The status shows the current role (slave), the master host, and the current status of the replicator.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DATASERVER(state=ONLINE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>The status of the dataserver service, which indicates the status of the underlying database service.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CONNECTIONS(created=0, active=0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Tungsten Connector service, showing the number of connections have been created on this service, and the number that are currently active.</td>
</tr>
</tbody>
</table>

The main service status output, as provided by `ls` at the top level, provides a quick overview of the overall status of the dataservice. More detailed information on each service, and the current status of the individual services can be monitored and managed through `cctrl`.

### 4.1.1. Latency or Relative Latency Display

Continuent Tungsten can operate using either absolute or relative latency. The two are distinguished according to how the difference between transaction commit times are handled:

- **Absolute latency** — (default) is the difference between when a transaction was applied to a slave and when the transaction was originally applied to the master.

- **Relative latency** — is the difference between now and when the last transaction was written to the slave.

Absolute latency indicates the difference between transaction times, but, may also provide a misleading impression of the cluster state if there are large transactions being applied, or if the slave has stopped or become ‘stuck’ due to a transient failure. This is because absolute latency shows the time difference between transactions. If a transaction takes 5 or 10 seconds to apply, the absolute latency will only display the difference between when the transaction was written, and only after this has occurred on both the master and the slave. The actual time difference between these may be less than a second, even though the transaction took 10 seconds to succeed.

Relative latency shows the time difference between the last transaction committed and the current time, hence if the transaction takes a considerable time to be applied, the relative latency will increase up until the transaction has finally been committed. If the relative latency increases and continues to increase, it may indicate a lagging or even failed slave.

To enable relative latency, the cluster must have been deployed, or updated, using the `--use-relative-latency=true` option to `tpm`. Once enabled, the following operational activities change:

- The output of `SHOW SLAVE STATUS` when connected to MySQL through a connector will be updated so that the `Seconds_Behind_Master` field shows the relative, rather than absolute, latency. For example, in a cluster where relative latency is enabled, but no transactions are occurring, the output will show an increasing value:

```sql
mysql> show slave status
+---------------------------+----------+
| Seconds_Behind_Master    | 0        |
|                          |         |
| 1 row in set (0.01 sec)  |         |
```

```sql
mysql> show slave status
+---------------------------+----------+
| Seconds_Behind_Master    | 7        |
|                          |         |
| 1 row in set (0.01 sec)  |         |
```

```sql
mysql> show slave status
+---------------------------+----------+
| Seconds_Behind_Master    | 38       |
|                          |         |
| 1 row in set (0.01 sec)  |         |
```
• `cctrl` will output an additional field, relative, showing the relative latency value against the standard latency value. This can be seen in the example below:

```plaintext
[LOGICAL] /alpha > ls

COORDINATOR[host1:AUTOMATIC:ONLINE]

ROUTERS:
  connector@host1[6189](ONLINE, created=1, active=0)
  connector@host2[14253](ONLINE, created=3, active=2)
  connector@host3[2419](ONLINE, created=1, active=0)

DATASOURCES:
  host1(master:ONLINE, progress=5, THL latency=1.008, relative=144.636)
  host2(slave:ONLINE, progress=5, latency=0.000, relative=144.638)
  host3(slave:ONLINE, progress=5, latency=5.938, relative=144.620)

• The Tungsten Connector will use the value when the `maxAppliedLatency` option is used in the connection string to determine whether to route a connection to a master or a slave.

For example, when running a script that sends a heartbeat, and then connects through a connector, the connection will be routed first to the slave, and then to the master:

```bash
echo "cluster heartbeat" | cctrl
sleep 1
mysql -utungsten_testing -pprivate --port=9999 --host=`hostname` \
  mysql@maxAppliedLatency=20?qos=RO_RELAXED -e"select 1;tungsten connection status;"
sleep 21
mysql -utungsten_testing -pprivate --port=9999 --host=`hostname` \
  mysql@maxAppliedLatency=20?qos=RO_RELAXED -e"select 1;tungsten connection status;"
```

The output of the execution of the script shows the slave and then master connections:
4.1.2. Getting Detailed Information

Detailed information about the individual nodes, datasources and services within the dataservice can be obtained by using the hierarchical structure of the dataservice as presented through `cctrl`. By using the `-l` command-line option detailed information can be obtained about any object. For example, getting the detailed listing of a specific host produces the following:

```plaintext
cctrl /alpha &gt; ls -l host1
```

**COORDINATOR:[host1:AUTOMATIC:ONLINE]**

**DATASOURCES:**

```plaintext
host1(master:ONLINE, progress=154146, THL latency=0.390)
```

**ROUTERS:**

```plaintext
CONNECTOR@host3[16117](ONLINE, created=0, active=0)
```

**Operations Guide**

By using the `-l` command-line option detailed information can be obtained about the structure of the dataservice as presented through `cctrl`. Detailed information about the individual nodes, datasources and services within the dataservice can be obtained by using the hierarchical structure of the dataservice as presented through `cctrl`. By using the `-l` command-line option detailed information can be obtained about any object. For example, getting the detailed listing of a specific host produces the following:

```plaintext
cctrl /alpha &gt; ls -l host1
```
The information output is very detailed and provides a summary of all the configuration and status information for the given host. The connector information shows connectors made to each configured datasource by each connector service. The datasource section shows detailed information on the datasource and replicator services. The output from the replicator service is equivalent to that output by `trepctl`.

### 4.1.3. Understanding Datasource Roles

All datasources within a dataservice have a specific role within the dataservice. The **master** role is one that provides a source of replication information, and a slave one that receives that information.

<table>
<thead>
<tr>
<th>Role</th>
<th>Supplies Replication Data</th>
<th>Receives Replication Data</th>
<th>Load Balancing</th>
<th>Failover</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master [89]</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Slave [89]</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Standby [89]</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Archive [89]</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

More detailed information for each role:

- **master [89]**

  A datasource in a **master [89]** role is providing a source for replication information to other datasources in the dataservice and is able to provide both read and write connections for applications.

- **slave [89]**

  A **slave [89]** datasource is receiving data from a **master [89]** and having that replicated data applied by Continuent Tungsten. Slaves are used for read-only operations by applications.

- **standby [89]**

  A **standby [89]** datasource receives replication data, but is never chosen by the connector to act as a read source by application clients. Standby datasources are therefore kept up to date with replication, but not used for load balancing.

  When a failover occurs, a **standby [89]** datasource can be enabled as a standard **slave [89]** and included in load-balanced operations.

- **archive [89]**
An archive datasource can be used to provide an active (up to date) copy of the data, without the datasource being used in the event of a failover. This can be useful for providing backup support, offline querying outside of the normal dataservice operations, or auditing purposes.

4.1.4. Understanding Datasource States

All datasources will be in one of a number of states that indicate their current operational status.

4.1.4.1. ONLINE State

A datasource in the ONLINE state is considered to be operating normally, with replication, connector and other traffic being handled as normal.

4.1.4.2. SHUNNED State

A SHUNNED datasource implies that the datasource is OFFLINE. Unlike the OFFLINE state, a SHUNNED datasource is not automatically recovered. A datasource in a SHUNNED state is not connected or actively part of the dataservice. Individual services can be reconfigured and restarted. The operating system and any other maintenance to be performed can be carried out while a host is in the SHUNNED state without affecting the other members of the dataservice.

Datasources can be manually or automatically shunned. The current reason for the SHUNNED state is indicated in the status output. For example, in the sample below, the node host3 was manually shunned for maintenance reasons:

```
+----------------------------------------------------------------------------+
|host3(slave:SHUNNED(MANUALLY-SHUNNED), progress=157454, latency=1.000)      |
|STATUS [SHUNNED] [2013/05/14 05:12:52 PM BST]                               |
+----------------------------------------------------------------------------+
```

4.1.4.3. OFFLINE State

A datasource in the OFFLINE does not accept connections through the connector for either reads or writes.

When the dataservice is in the AUTOMATIC policy mode, a datasource in the OFFLINE state is automatically recovered and placed into the ONLINE state. If this operation fails, the datasource remains in the OFFLINE state.

When the dataservice is in MAINTENANCE or MANUAL policy mode, the datasource will remain in the OFFLINE state until the datasource is explicitly switched to the ONLINE state.

4.1.4.4. FAILED State

When a datasource fails, for example when a failure in one of the services for the datasource stops responding or fails, the datasource will be placed into the FAILED state. In the example below, the underlying dataserver has failed:

```
+----------------------------------------------------------------------------+
|host3(slave:FAILED(DATASERVER 'host3@alpha' STOPPED), progress=154146, latency=31.419) |
|STATUS [CRITICAL] [2013/05/10 11:51:42 PM BST]                                 |
|REASON [DATASERVER 'host3@alpha' STOPPED]                                      |
+----------------------------------------------------------------------------+
```

For a FAILED datasource, the recover command within cctrl can be used to attempt to recover the datasource to the operational state. If this fails, the underlying fault must be identified and addressed before the datasource is recovered.

4.1.5. Changing Datasource States

Changing the status of a service is required either when the dataservice needs to be reconfigured, the topology altered, or when performing system maintenance.

The datasource status can be changed by using the datasource command, which accepts the datasource name and a sub-command:

```
datasource DATASOURCENAME SUBCOMMAND
```
For example, to shun the node `host1`:

```
[LOGICAL:EXPERT] /alpha > datasource host1 shun
```

For detailed operations for different subcommands, see the following sections.

### 4.1.5.1. Shunning a Datasource

Shunning a datasource identifies the source as unavailable; a shunned slave will not be used during a failover or switch operation.

Datasources can be automatically or manually shunned:

- **Automatic** shunning occurs when the dataservice is in **AUTOMATIC** policy mode, and the datasource has become unresponsive or fails. For example, when a master fails, an automatic switch to a new master is performed, and the old master is shunned.

- **Manual** shunning occurs when the `shun` command is given to a datasource. Manual shunning can be used to set a datasource into a state that allows for maintenance and management operations to be performed on the datasource.

To manually shun the datasource:

```
[LOGICAL:EXPERT] /alpha > datasource host3 shun
```

Once shunned, the connector will stop using the datasource. The status can be checked using `ls`:

```
+----------------------------------------------------------------------------+
|host3(slave:SHUNNED(MANUALLY-SHUNNED), progress=157454, latency=1.000)      |
|STATUS [SHUNNED] [2013/05/14 05:24:41 PM BST]                               |
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=slave, master=host2, state=ONLINE)                        |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+
```

**Important**

Shunning a datasource does not stop the replicator; replication will continue on a shunned datasource until the replicator service is explicitly placed into the offline state.

The level of the shunning is reported in the status as a manual operation. A manually shunned datasource can be enabled using the `datasource recover` command, see Section 4.1.5.2, “Recover a Datasource”.

### 4.1.5.2. Recover a Datasource

The `datasource recover` command is a deeper operation that performs a number of operations to get the datasource back into the operational state. When used, the `datasource recover` command performs the following operations:

- Restarts failed or stopped services

- Changes the datasource configuration so that it is configured as a master or slave. For example, an automatically failed master will be reconfigured to operate as a slave to the current master.

- Restarts the replicator service in the slave or master role as appropriate

In all cases, the `datasource recover` command should be used if a datasource is offline or shunned, and it can be used at all times to get a datasource back in to operational state within the cluster. In essence, `recover` performs the same operations automatically as would be performed manually to get the node into the right state.

```
[LOGICAL:EXPERT] /alpha > datasource host3 recover
VERIFYING THAT WE CAN CONNECT TO DATA SERVER 'host3'
DATA SERVER 'host3' IS NOW AVAILABLE FOR CONNECTIONS
RECOVERING 'host3@alpha' TO A SLAVE USING 'host1@alpha' AS THE MASTER
DataSource 'host3' is now OFFLINE
RECOVERY OF 'host3@alpha' WAS SUCCESSFUL
```

During the recovery process, the node will be checked, replication reconfigured, and the node brought back in to active service. If this process fails because the databases and replication states are out of sync and cannot be recovered, Continuent Tungsten may advise that a backup of another datasource and recovery to this datasource is performed. For more information on restoring from backups, see Section 4.10, “Restoring a Backup”.
4.1.5.3. Offline a Datasource

A datasource can be explicitly placed into offline mode. In offline mode, client applications connections to datasources are paused. When switching to offline mode existing connections are given a five-second grace period to complete their operations before being forced to disconnect. Replicator operation is not affected.

To set a datasource offline:

```
[LOGICAL:EXPERT] /alpha > datasource host3 offline
DataSource 'host3@alpha' is now OFFLINE
```

If the dataservice is in AUTOMATIC policy mode, and there are no other faults in the datasource, it will automatically be placed into ONLINE mode. To set a datasource offline the dataservice must be in MAINTENANCE or MANUAL policy modes.

4.1.5.4. Mark a Datasource as Standby

A standby datasource receives replication data, but are not part of the load-balancing provided by Tungsten Connector. In the event of a failover situation, a standby datasource will be enabled within the cluster as a slave. Because the standby datasource is up to date with respect to the replication of data, this process is instantaneous. The connector will be updated, and the new slave will operate as a read-only datasource.

To configure a datasource as a standby:

```
[LOGICAL:EXPERT] /alpha > datasource host3 standby
Datasource 'host3' now has role 'standby'
```

To clear the standby state:

```
[LOGICAL:EXPERT] /alpha > datasource host3 clear standby
Datasource 'host3' now has role 'slave'
```

4.1.5.5. Mark a Datasource as Archive

An archive datasource receives replication data. It is excluded from failover switches and will not be used as a master in the event of a failure. To mark a datasource as an archive datasource:

```
[LOGICAL:EXPERT] /alpha > datasource host3 set archive
```

To remove the archive role:

```
[LOGICAL:EXPERT] /alpha > datasource host3 clear archive
```

The archive role is a temporary requirement, and will not survive a re-install or upgrade.

4.1.6. Datasource Statuses

In addition to the overall state, all datasources have a specific status that indicates the current health and operation, rather than the configured state for that datasource. For example, a datasource can be in the online state, but have a diminished status if there is a recoverable problem with one of the datasource components.

- **OK**
  
  The OK status indicates that the datasource is currently operating correctly.

- **DIMINISHED**
  
  A diminished status indicates that there is a problem with one of the dataservice services which is causing a reduced level of expected service. For example, in the sample output below, the reason is indicated as a stopped replicator service.

```
+----------------------------------------------------------------------------+
|host1(master:ONLINE)                                                        |
|STATUS [DIMINISHED] [2013/05/11 12:38:33 AM BST]                           |
|REASON(REPLICATOR STOPPED)                                                 |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                    |
|  REPLICATOR(state=STOPPED)                                                 |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=195, active=0)                                        |
+----------------------------------------------------------------------------+
```

The underlying service fault should be fixed and the status rechecked.
4.1.7. Datasource States and Policy Mode Interactions

States can be explicit set through `cctrl` command, however, depending on the current policy mode, the actual status set may be different from that initially set. For example, when shunning a datasource, the datasource will immediately go into `SHUNNED` state.

```
[LOGICAL:EXPERT] /alpha > datasource host3 shun
DataSource 'host3' set to SHUNNED
```

Figure 4.1. Sequence: Shunning a Datasource

To bring the datasource back into operation, it must be brought back using the `recover` command:

```
[LOGICAL:EXPERT] /alpha > datasource host3 recover
DataSource 'host3' is now OFFLINE
```

The `datasource recover` command performs whatever steps are necessary to bring the datasource back into operation within the dataservice. Even for manually shunned datasources, there may be additional configuration or recovery steps required.

If the dataservice policy mode is `MANUAL` or `MAINTENANCE` modes, the datasource remains in the `OFFLINE` state until manually put `ONLINE`.

4.2. Policy Modes

The dataservice operates using a policy mode, which configures how the dataservice management system responds to different events and operations within the dataservice. The policy mode can be set at will and enables maintenance and administration to be carried out without triggering the automatic failure and recovery procedures for operations that would otherwise trigger an automated response.

The procedure for how these operations are carried out are defined through a series of rules, with different policies applying different sets of the individual rules. The setting of the policy mode is dataservice-wide and instantaneous.

<table>
<thead>
<tr>
<th>Ruleset</th>
<th>Policy Mode</th>
<th>Manual</th>
<th>Maintenance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitoring</td>
<td>Automatic</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
The individual policy modes are described below:

- **AUTOMATIC** Policy Mode
  
  In automatic mode, the following operations and status changes happen automatically, managed by the coordinator:
  
  - Failed slave datasources are automatically marked as failed, temporarily removed from the dataservice, with application connections redirect to the other nodes in the dataservice. When the datasource becomes available, the node is automatically recovered to the dataservice.
  
  - Failed master datasources are automatically shunned and switched to the most up to date slave within the dataservice, which becomes the master and remaining slaves point to the newly promoted master.
  
  **Note**

  Automatic policy mode operates within a single dataservice only. Within a composite dataservice there is no automatic failover.

- **MANUAL** Policy Mode
  
  In the Manual policy mode, the dataservice identifies and isolates datasources when they fail, but automatic failover (for master datasources) and recovery is disabled.

- **MAINTENANCE** Policy Mode
  
  In Maintenance policy mode all rules are disabled. Maintenance mode should be used when performing datasource or host maintenance that would otherwise trigger an automated fencing or recovery process.

  Maintenance mode should be used when administration or maintenance is required on the datasource, software, or operating system.

### 4.2.1. Setting Policy Modes

To set the policy, use the `set` command with the policy option. For example, to switch the current dataservice policy mode to manual:

```plaintext
[LOGICAL:EXPERT] /alpha > set policy manual
policy mode is now MANUAL
```

Policy mode changes are global, affecting the operation of all the members of the dataservice.

The current policy mode is shown when running `ls` within `cctrl`, see Section 4.1, “Checking Dataservice Status”.

### 4.3. Switching Master Hosts

The master host within a dataservice can be switched, either automatically, or manually. Automatic switching occurs when the dataservice is in the **AUTOMATIC** policy mode, and a failure in the underlying datasource has been identified. The automatic process is designed to keep the dataservice running without requiring manual intervention.

Manual switching of the master can be performed during maintenance operations, for example during an upgrade or dataserver modification. In this situation, the master must be manually taken out of service, but without affecting the rest of the dataservice. By switching the master to another datasource in the dataservice, the original master can be put offline, or shunned, while maintenance occurs. Once the maintenance has been completed, the datasource can be re-enabled, and either remain as the a slave, or switched back as the master datasource.

Switching a datasource, whether automatically or manually, occurs while the dataservice is running, and without affecting the operation of the dataservice as a whole. Client application connections through Tungsten Connector are automatically reassigned to the datasources in the dataservice, and application operation will be unaffected by the change. Switching the datasource manually requires a single command that performs all of the required steps, monitoring and managing the switch process.

Switching the master, manually or automatically, performs the following steps within the dataservice:

1. Set the master node to offline state. New connections to the master are rejected, and writes to the master are stopped.
2. On the slave that will be promoted, switch the datasource offline. New connections are rejected, stopping reads on this slave.

3. Kill any outstanding client connections to the master data source, except those belonging to the tungsten account.

4. Send a heartbeat transaction between the master and the slave, and wait until this transaction has been received. Once received, the THL on master and slave are up to date.

5. Perform the switch:
   - Configure all remaining replicators offline
   - Configure the selected slave as the new master.
   - Set the new master to the online state.
   - New connections to the master are permitted.

6. Configure the remaining slaves to use the new master as the master datasource.

7. Update the connector configurations and enable client connections to connect to the masters and slaves.

The switching process is monitoring by Continuent Tungsten, and if the process fails, either due to a timeout or a recoverable error occurs, the switch operation is rolled back, returning the dataservice to the original configuration. This ensures that the dataservice remains operational. In some circumstances, when performing a manual switch, the command may need to be repeated to ensure the requested switch operation completes.

The process takes a finite amount of time to complete, and the exact timing and duration will depend on the state, health, and database activity on the dataservice. The actual time taken will depend on how up to date the slave being promoted is compared to the master. The switch will take place regardless of the current status after a configurable delay period. For more information, see ???.

### 4.3.1. Automatic Master Switch

When the dataservice policy mode is **AUTOMATIC**, the dataservice will automatically switch the master host when the existing master is identified as having failed or become unavailable.

For example, when the master host `host1` becomes unavailable because of a network problem, the dataservice automatically switches to `host2`. The dataservice status is updated accordingly, showing the automatically shunned `host1`:

```
[LOGICAL:EXPERT] /alpha > ls
COORDINATOR[host3:AUTOMATIC:ONLINE]

ROUTERS:
+----------------------------------------------------------------------------+
|connector@host2[28116](ONLINE, created=0, active=0)                     |
|connector@host3[1533](ONLINE, created=0, active=0)                      |
+----------------------------------------------------------------------------+

DATASOURCES:
+----------------------------------------------------------------------------+
|host1(master:SHUNNED(FAILED-OVER-TO-host2))                              |
|STATUS [SHUNNED] [2013/05/14 12:18:54 PM BST]                           |
| MANAGER(state=STOPPED)                                                   |
| REPLICA(state=STATUS NOT AVAILABLE)                                      |
| DATASERVER(state=ONLINE)                                                 |
| CONNECTIONS(created=0, active=0)                                         |
+----------------------------------------------------------------------------+

+----------------------------------------------------------------------------+
|host2(master:ONLINE, progress=186325, THL latency=0.606)                |
|STATUS [OK] [2013/05/14 12:46:55 PM BST]                                 |
| MANAGER(state=ONLINE)                                                   |
| REPLICA(role=master, state=ONLINE)                                      |
| DATASERVER(state=ONLINE)                                                 |
| CONNECTIONS(created=0, active=0)                                         |
+----------------------------------------------------------------------------+
```

The status for the original master (`host1`) identifies the datasource as shunned, and indicates which datasource was promoted to the master in the `FAILED-OVER-TO-host2`.

A automatic failover can be triggered by using the `datasource fail` command:

```
[LOGICAL:EXPERT] /alpha > datasource host1 fail
```

---

95
This triggers the automatic failover sequence, and simulates what would happen if the specified host failed.

If `host1` becomes available again, the datasource is not automatically added back to the dataservice, but must be explicitly re-added to the dataservice. The status of the dataservice once `host1` returns is shown below:

```
[LOGICAL:EXPERT] /alpha > ls
COORDINATOR[host3:AUTOMATIC:ONLINE]
ROUTERS:
+----------------------------------------------------------------------------+
|connector@host1[19869](ONLINE, created=0, active=0)                         |
|connector@host2[28116](ONLINE, created=0, active=0)                         |
|connector@host3[1533](ONLINE, created=0, active=0)                          |
+----------------------------------------------------------------------------+
DATASOURCES:
+----------------------------------------------------------------------------+
|host1(master:SHUNNED(FAILED-OVER-TO-host2), progress=156323, THL            |
|latency=0.317)                                                              |
|STATUS [SHUNNED] [2013/05/14 12:30:21 PM BST]                               |
+----------------------------------------------------------------------------+
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=master, state=ONLINE)                                     |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
+----------------------------------------------------------------------------+
```

Because `host1` was previously the master, the `datasource recover` command verifies that the server is available, configures the node as a slave of the newly promoted master, and re-enables the services:

```
[LOGICAL:EXPERT] /alpha > datasource host1 recover
VERIFYING THAT WE CAN CONNECT TO DATA SERVER 'host1'
DATA SERVER 'host1' IS NOW AVAILABLE FOR CONNECTIONS
RECOVERING 'host1@alpha' TO A SLAVE USING 'host2@alpha' AS THE MASTER
SETTING THE ROLE OF DATASOURCE 'host1@alpha' FROM 'master' TO 'slave'
RECOVERY OF 'host1@alpha' WAS SUCCESSFUL
```

If the command is successful, then the node should be up and running as a slave of the new master.

The recovery process can fail if the THL data and dataserver contents do not match, for example when statements have been executed on a slave. For information on recovering from failures that `recover` cannot fix, see Section 4.5.1.3, "Slave Datasource Extended Recovery".

### 4.3.2. Manual Master Switch

In a single data service dataservice configuration, the master can be switched between nodes within the dataservice manually using `cctrl`. The `switch` command performs the switch operation, annotating the progress.

```
[LOGICAL:EXPERT] /alpha > switch
SELECTED SLAVE: host2@alpha
PURGE REMAINING ACTIVE SESSIONS ON CURRENT MASTER 'host1@alpha'
PURGED A TOTAL OF 0 ACTIVE SESSIONS ON MASTER 'host1@alpha'
FLUSH TRANSACTIONS ON CURRENT MASTER 'host1@alpha'
PUT THE NEW MASTER 'host2@alpha' ONLINE
PUT THE PRIOR MASTER 'host1@alpha' ONLINE AS A SLAVE
RECONFIGURING SLAVE 'host3@alpha' TO POINT TO NEW MASTER 'host2@alpha'
SWITCH TO 'host2@alpha' WAS SUCCESSFUL
```

By default, `switch` chooses the most up to date slave within the dataservice ( `host2` in the above example), but an explicit slave can also be selected:

```
[LOGICAL:EXPERT] /alpha > switch to host3
SELECTED SLAVE: host3@alpha
PURGE REMAINING ACTIVE SESSIONS ON CURRENT MASTER 'host2@alpha'
PURGED A TOTAL OF 0 ACTIVE SESSIONS ON MASTER 'host2@alpha'
FLUSH TRANSACTIONS ON CURRENT MASTER 'host2@alpha'
PUT THE NEW MASTER 'host3@alpha' ONLINE
PUT THE PRIOR MASTER 'host2@alpha' ONLINE AS A SLAVE
RECONFIGURING SLAVE 'host3@alpha' TO POINT TO NEW MASTER 'host3@alpha'
SWITCH TO 'host3@alpha' WAS SUCCESSFUL
```

With the previous example, the switch occurred specifically to the node `host3`.

### 4.4. Replicator Fencing
Continuent Tungsten can be configured to handle failures during replication automatically and to fence the failure so that the issues do not lead to issues with the rest of the cluster, which may lead to problems with applications operating against the cluster. By default, the cluster is designed to take no specific action aside from indicating and registering the replicator so that the node will be identified as being within the **DIMINISHED** or **CRITICAL** state.

This behavior can be changed so that the failed replicator failure is fenced, with configuration operating on either the master, or slave replicators. When fencing has been enabled, the node will be placed into either the **OFFLINE** state if the node is a slave or a failover will occur if the node is a master.

### 4.4.1. Fencing a Slave Node Due to a Replication Fault

If the replicator should be placed into the **OFFLINE** state when replicator stops or raises an error, the following option can be set through `tpm` on the cluster configuration to set the `policy.fence.slaveReplicator` to true:

```shell
> tpm update alpha --property=policy.fence.slaveReplicator=true
```

The delay before the fencing operation takes place can be configured using the `policy.fence.slaveReplicator.threshold` parameter, which configures the delay before taking action, with the value multiplied by 10. For example, a setting of 6 implies a delay of 60 seconds. The delay enables transient errors, such as network failures, to be effectively managed without automatically fencing the slave.

```shell
> tpm update alpha --property=policy.fence.slaveReplicator.threshold=6
```

Once a slave has been fenced, the state will automatically be cleared when the replicator returns to the **ONLINE** state. Once this has been identified, the node will be placed in the **ONLINE** state.

### 4.4.2. Fencing Master Replicators

In the event of a master replicator failure, the fencing operation places the datasource into the **FAILED** state, triggering an automatic failover (see Section 4.3.1, “Automatic Master Switch”). Because this triggers a failover in the event of fencing the replicator, the configuration should only be enabled if it critical for your business that replication errors/stoppages should trigger a significant operation as failover.

To enable fencing of the master node due to replication faults, use the `policy.fence.masterReplicator` configuration property when configuring the cluster:

```shell
> tpm update alpha --property=policy.fence.masterReplicator=true
```

The delay before the fencing operation takes place can be configured using the `policy.fence.masterReplicator.threshold` property. The default value is 3, or 30 seconds.

```shell
> tpm update alpha --property=policy.fence.masterReplicator.threshold=6
```

When the replicator is identified as available, the master datasource is not placed back into the online state. Instead, the failed datasource and must be explicitly recovered using the `recover` or `datasource host recover` commands.

### 4.5. Datasource Recovery Steps

When a datasource within the dataservice fails, the exact response by the dataservice is dependent on the dataservice policy mode. Different policy modes either cope with the failure or recovery process automatically, or a prescribed sequence must be followed.

Recovery can normally be achieved by following these basic steps:

- Use the `recover` command

  The `recover` command performs a number of steps to try and return the datasource to the operational state, but works only if there is an existing master within the current configuration. Operations conducted automatically include slave recovery, and reconfiguring roles. For example:

```
[LOGICAL] /alpha > recover

FOUND PHYSICAL Datasource TO RECOVER: 'host2@alpha'
VERIFYING THAT WE CAN CONNECT TO DATA SERVER 'host2'
DATA SERVER 'host2' IS NOW AVAILABLE FOR CONNECTIONS
RECOVERING 'host2@alpha' TO A SLAVE USING 'host3@alpha' AS THE MASTER
DataSource 'host2' is now OFFLINE
RECOVERY OF Datasource 'alpha' SUCCEEDED
FOUND PHYSICAL Datasource TO RECOVER: 'host1@alpha'
VERIFYING THAT WE CAN CONNECT TO DATA SERVER 'host1'
DATA SERVER 'host1' IS NOW AVAILABLE FOR CONNECTIONS
RECOVERING 'host1@alpha' TO A SLAVE USING 'host3@alpha' AS THE MASTER
DataSource 'host1' is now OFFLINE
RECOVERY OF Datasource 'alpha' SUCCEEDED
```
98

**RECOVERED 2 DATA SOURCES IN SERVICE ‘alpha’**

- **Slave failure, Master still available**

  Use the `recover` to bring all slaves back into operation. To bring a single slave, use the `datasource recover`:

  ```
  [LOGICAL:EXPERT] /alpha > datasource host1 recover
  VERIFYING THAT WE CAN CONNECT TO DATA SERVER ‘host1’
  DATA SERVER ‘host1’ IS NOW AVAILABLE FOR CONNECTIONS
  RECOVERING ‘host1@alpha’ TO A SLAVE USING ‘host2@alpha’ AS THE MASTER
  RECOVERY OF ‘host1@alpha’ WAS SUCCESSFUL
  ```

  If recovery of the slave fails with this method, you can try more advanced solutions for getting your slave(s) working, including reprovisioning from another slave.

  For more info, see Section 4.5.1, “Recover a failed slave”.

- **Master failure**

  If the most up to date master can be identified, use the `recover using` command to set the new master and recover the remaining slaves. If this does not work, use the `set master` command and then use the `recover` command to bring back as many possible slaves, and then use a backup/restore operation to bring any other slaves back into operation, or use the `tungsten_provision_slave` command. For more information, see Section 4.5.2, “Recover a failed master”.

  A summary of these different scenarios and steps is provided in the following table:

<table>
<thead>
<tr>
<th>Policy Mode</th>
<th>Scenario</th>
<th>Datasource State</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTOMATIC</td>
<td>Master Failure</td>
<td>master:SHUNNED(FAILED-OVER-TO-host2)</td>
<td>Section 4.5.2, “Recover a failed master”</td>
</tr>
<tr>
<td>AUTOMATIC</td>
<td>Master Recovery</td>
<td>master:SHUNNED(FAILED-OVER-TO-host2)</td>
<td>Section 4.5.2, “Recover a shunned master”</td>
</tr>
<tr>
<td>MANUAL</td>
<td>Master Failure</td>
<td>master:FAILED(NODE ‘host1’ IS UNREACHABLE))</td>
<td>Section 4.5.2.4, “Failing over a master”</td>
</tr>
<tr>
<td>MANUAL</td>
<td>Master Recovery</td>
<td>master:SHUNNED(FAILED-OVER-TO-host2)</td>
<td>Section 4.5.2.2, “Recover a shunned master”</td>
</tr>
<tr>
<td>MANUAL</td>
<td>Slave Failure</td>
<td>slave:FAILED(NODE ‘host1’ IS UNREACHABLE)</td>
<td>Automatically removed from service</td>
</tr>
<tr>
<td>MANUAL</td>
<td>Slave Recovery</td>
<td>slave:FAILED(NODE ‘host1’ IS UNREACHABLE)</td>
<td>Section 4.5.1, “Recover a failed slave”</td>
</tr>
<tr>
<td>Maintenance</td>
<td>Master Failure</td>
<td>Use Section 4.5.2.4, “Failing over a master”</td>
<td>to promote a different slave</td>
</tr>
<tr>
<td>Maintenance</td>
<td>Master Recovery</td>
<td></td>
<td>Section 4.5.2.3, “Manually Failing over a Master in MAINTENANCE policy mode”</td>
</tr>
<tr>
<td></td>
<td>Slave Failure</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Slave Recovery</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>Any</td>
<td>Slave Shunned</td>
<td>slave:SHUNNED(MANUALLY-SHUNNED)</td>
<td>Section 4.5.1, “Recover a failed slave”</td>
</tr>
<tr>
<td>Any</td>
<td>No Master</td>
<td>slave:SHUNNED(SHUNNED)</td>
<td>Section 4.5.2.1, “Recover when there are no masters”</td>
</tr>
</tbody>
</table>

### 4.5.1. Recover a failed slave

A slave that has failed but which has become available again can be recovered back into slave mode using the `recover` command:
The recover command will attempt to recover all the slave resources in the cluster, bringing them all online and back into service. The command operates on all shunned or failed slaves, and only works if there is an active master available.

To recover a single datasource back into the dataservice, use the explicit form:

```
[LOGICAL:EXPERT] /alpha > datasource host1 recover
```

In some cases, the datasource may show as **ONLINE** and the `recover` command does not bring the datasource online, particularly with the following error:

```
The datasource 'host1' is not FAILED or SHUNNED and cannot be recovered.
```

Checking the datasource status in `cctrl` the replicator service has failed, but the datasource shows as online:

```
+----------------------------------------------------------------------------+
| host1 (slave:ONLINE, progress=-1, latency=-1.000)                        |
| STATUS [OK] [2013/06/24 12:42:06 AM BST]                                |
+----------------------------------------------------------------------------+
| MANAGER(state=ONLINE)                                                    |
| REPLICATOR(role=slave, master=host1, state=SUSPECT)                     |
| DATASERVER(state=ONLINE)                                                 |
+----------------------------------------------------------------------------+
```

In this case, the datasource can be manually shunned, which will then enable the `recover` command to operate and bring the node back into operation.

### 4.5.1.1. Provision or Reprovision a Slave

In the event that you cannot get the slave to recover using the `datasource recover` command, you can re-provision the slave from another slave within your dataservice.

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:
NOTE >> Load the mysqldump file
NOTE >> Put the alpha replication service online
NOTE >> Clear THL and relay logs for the alpha replication service

The default backup service for the host will be used; mysqldump can be used by specifying the --mysqldump option.

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.

For more information on using tungsten_provision_slave see Section 7.10, "The tungsten_provision_slave Script".

4.5.1.2. Recover a slave from manually shunned state

A slave that has been manually shunned can be added back to the dataservice using the datasource recover command:

```
[LOGICAL:EXPERT] /alpha > datasource host3 recover
DataSource 'host3' is now OFFLINE
```

In AUTOMATIC policy mode, the slave will automatically be recovered from OFFLINE to ONLINE mode.

In MANUAL or MAINTENANCE policy mode, the datasource must be manually switched to the online state:

```
[LOGICAL:EXPERT] /alpha > datasource host3 online
Setting server for data source 'host3' to READ-ONLY
+----------------------------------------------------------------------------+
|host3                                                                       |
+----------------------------------------------------------------------------+
|Variable_name  Value                                                        |
|read_only  ON                                                               |
+----------------------------------------------------------------------------+
DataSource 'host3@alpha' is now ONLINE
```

4.5.1.3. Slave Datasource Extended Recovery

If the current slave will not recover, but the replicator state and sequence number are valid, the slave is pointing to the wrong master, or still mistakenly has the master role when it should be a slave, then the slave can be forced back into the slave state.

For example, in the output from ls in cctrl below, host2 is mistakenly identified as the master, even though host1 is correctly operating as the master.

```
COORDINATOR[host1:AUTOMATIC:ONLINE]

ROUTERS:

|connector@host1[1848](ONLINE, created=0, active=0)                          |
|connector@host2[4098](ONLINE, created=0, active=0)                          |
|connector@host3[4087](ONLINE, created=0, active=0)                          |

DATASOURCES:

|host1(master:ONLINE, progress=23, THL latency=0.198)                        |
|STATUS [OK] [2013/05/30 11:29:44 AM BST]                                   |
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=master, state=ONLINE)                                     |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |

|host2(slave:SHUNNED(MANUALLY-SHUNNED), progress=-1, latency=-1.000)         |
|STATUS [SHUNNED] [2013/05/30 11:23:15 AM BST]                               |
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=master, state=OFFLINE)                                    |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |

|host3(slave:ONLINE, progress=23, latency=178877.000)                        |
|STATUS [OK] [2013/05/30 11:33:15 AM BST]                                   |
|  MANAGER(state=ONLINE)                                                     |
|  REPLICATOR(role=slave, master=host1, state=ONLINE)                       |
|  DATASERVER(state=ONLINE)                                                  |
|  CONNECTIONS(created=0, active=0)                                          |
The datasource `host2` can be brought back online using this sequence:

1. Enable *force* mode:

   ```
   [LOGICAL:EXPERT] /alpha > set force true
   FORCE: true
   ```

2. Shun the datasource:

   ```
   [LOGICAL:EXPERT] /alpha > datasource host2 shun
   DataSource 'host2' set to SHUNNED
   ```

3. Switch the replicator offline:

   ```
   [LOGICAL:EXPERT] /alpha > replicator host2 offline
   Replicator 'host2' is now OFFLINE
   ```

4. Set the replicator to *slave* operation:

   ```
   [LOGICAL:EXPERT] /alpha > replicator host2 slave
   Replicator 'host2' is now a slave of replicator 'host1'
   ```

   In some instances you may need to explicitly specify which node is your master when you configure the slave; appending the master hostname to the command specifies the master host to use:

   ```
   [LOGICAL:EXPERT] /alpha > replicator host2 slave host1
   Replicator 'host2' is now a slave of replicator 'host1'
   ```

5. Switch the replicator service online:

   ```
   [LOGICAL:EXPERT] /alpha > replicator host2 online
   Replicator 'host2' is now ONLINE
   ```

6. Ensure the datasource is correctly configured as a slave:

   ```
   [LOGICAL:EXPERT] /alpha > datasource host2 slave
   Datasource 'host2' now has role 'slave'
   ```

7. Recover the slave back to the dataservice:

   ```
   [LOGICAL:EXPERT] /alpha > datasource host2 recover
   DataSource 'host2' is now OFFLINE
   ```

Datasource `host2` should now be back in the dataservice as a working datasource.

Similar processes can be used to force a datasource back into the *master* role if a switch or recover operation failed to set the role properly.

If the `recover` command fails, there are a number of solutions that may bring the dataservice back to the normal operational state. The exact method will depend on whether there are other active slaves (from which a backup can be taken) or recent backups of the slave are available, and the reasons for the original failure. Some potential solutions include:

- If there is a recent backup of the failed slave, restore the slave using that backup. The latest backup can be restored using Section 4.10, “Restoring a Backup”.
- If there is no recent backup, but have another slave from which you can recover the slave, the node should be rebuilt using the backup from another slave. See Section 4.10.3, “Restoring from Another Slave”.

### 4.5.2. Recover a failed master

When a master datasource is automatically failed over in *AUTOMATIC* policy mode, the datasource can be brought back into the dataservice as a slave by using the `recover` command:

```
[LOGICAL:EXPERT] /alpha > datasource host1 recover
VERIFYING THAT WE CAN CONNECT TO DATA SERVER 'host1'
DATA SERVER 'host1' IS NOW AVAILABLE FOR CONNECTIONS
RECOVERING 'host1@alpha' TO A SLAVE USING 'host2@alpha' AS THE MASTER
SETTING THE ROLE OF DATASOURCE 'host1@alpha' FROM 'master' TO 'slave'
RECOVERY OF 'host1@alpha' WAS SUCCESSFUL
```

The recovered datasource will be added back to the dataservice as a slave.
4.5.2.1. Recover when there are no masters

When there are no masters available, due to a failover of a master, or multiple host failure there are two options available. The first is to use the recover master using, which sets the master to the specified host, and tries to automatically recover all the remaining nodes in the dataservice. The second is to manually set the master host, and recover the remainder of the dataservices manually.

- Using recover master using

  Warning

  This command should only be used in urgent scenarios where the most up to date master can be identified. If there are multiple failures or mismatches between masters and slaves, the command may not be able to recover all services, but will always result in an active master being configured.

  This command performs two distinct actions, first it calls set master to select the new master, and then it calls datasource recover on each of the remaining slaves. This attempts to recover the entire dataservice by switching the master and reconfiguring the slaves to work with the new master.

  To use, first you should examine the state of the dataservice and choose which datasource is the most up to date or canonical. For example, within the following output, each datasource has the same sequence number, so any datasource could potentially be used as the master:

  ```
  [LOGICAL] /alpha > ls
  COORDINATOR[host1:AUTOMATIC:ONLINE]
  ROUTERS:
  [connector@host1[18450](ONLINE, created=0, active=0)]
  [connector@host2[8877](ONLINE, created=0, active=0)]
  [connector@host3[8895](ONLINE, created=0, active=0)]
  DATASOURCES:
  [host1(master:SHUNNED(Failsafe after Shunned by fail-safe procedure),
  progress=17, THL latency=0.565)]
  STATUS [OK] [2013/11/04 04:39:28 PM GMT]
  [MANAGER(state=ONLINE)]
  [REPLICATOR(role=master, state=ONLINE)]
  [DATASERVER(state=ONLINE)]
  [CONNECTIONS(created=0, active=0)]
  [host2(slave:SHUNNED(Failsafe after Shunned by fail-safe procedure),
  progress=17, latency=1.003)]
  STATUS [OK] [2013/11/04 04:39:51 PM GMT]
  [MANAGER(state=ONLINE)]
  [REPLICATOR(role=slave, master=host1, state=ONLINE)]
  [DATASERVER(state=ONLINE)]
  [CONNECTIONS(created=0, active=0)]
  [host3(slave:SHUNNED(Failsafe after Shunned by fail-safe procedure),
  progress=17, latency=1.273)]
  STATUS [OK] [2013/10/26 06:30:26 PM BST]
  [MANAGER(state=ONLINE)]
  [REPLICATOR(role=slave, master=host1, state=ONLINE)]
  [DATASERVER(state=ONLINE)]
  [CONNECTIONS(created=0, active=0)]
  ```

  Once a host has been chosen, call the recover master using command specifying the full servicename and hostname of the chosen datasource:

  ```
  [LOGICAL] /alpha > recover master using alpha/host1
  ```

  This command is generally meant to help in the recovery of a data service that has data sources shunned do to a fail-safe shutdown of the service or under other circumstances where you wish to force a specific data source to become the primary. Be forewarned that if you do not exercise care when using this command you may lose data permanently or otherwise make your data service unusable.

  Do you want to continue? (y/n)> y

  DATA SERVICE 'alpha' DOES NOT HAVE AN ACTIVE PRIMARY. CAN PROCEED WITH 'RECOVER USING'
VERIFYING THAT WE CAN CONNECT TO DATA SERVER 'host1'
DATA SERVER 'host1' IS NOW AVAILABLE FOR CONNECTIONS
DataSource 'host1' is now OFFLINE
DATASOURCE 'host1@alpha' IS NOW A MASTER
FOUND PHYSICAL DATASOURCE TO RECOVER: 'host2@alpha'
VERIFYING THAT WE CAN CONNECT TO DATA SERVER 'host2'
DATA SERVER 'host2' IS NOW AVAILABLE FOR CONNECTIONS
RECOVERING 'host2@alpha' TO A SLAVE USING 'host1@alpha' AS THE MASTER
DataSource 'host2' is now OFFLINE
RECOVERY OF DATA SERVICE 'alpha' SUCCEEDED
FOUND PHYSICAL DATASOURCE TO RECOVER: 'host3@alpha'
VERIFYING THAT WE CAN CONNECT TO DATA SERVER 'host3'
DATA SERVER 'host3' IS NOW AVAILABLE FOR CONNECTIONS
RECOVERING 'host3@alpha' TO A SLAVE USING 'host1@alpha' AS THE MASTER
DataSource 'host3' is now OFFLINE
RECOVERY OF DATA SERVICE 'alpha' SUCCEEDED
RECOVERED 2 DATA SOURCES IN SERVICE 'alpha'

You will be prompted to ensure that you wish to choose the selected host as the new master. `cctrl` then proceeds to set the new master, and recover the remaining slaves.

If this operation fails, you can try the manual process, using `set master` and proceeding to recover each slave manually.

• Using `set master`

The `set master` command forcibly sets the master to the specified host. It should only be used in the situation where no master is currently available within the dataservice, and recovery has failed. This command performs only one operation, and that is to explicitly set the new master to the specified host.

**Warning**

Using `set master` is an expert level command and may lead to data loss if the wrong master is used. Because of this, `cctrl` must be forced to execute the command by using `set force true` . The command will not be executed otherwise.

To use the command, pick the most up to date master, or the host that you want to use as the master within your dataservice, then issue the command:

```bash
[LOGICAL] /alpha > set master host3
VERIFYING THAT WE CAN CONNECT TO DATA SERVER 'host3'
DATA SERVER 'host3' IS NOW AVAILABLE FOR CONNECTIONS
DataSource 'host3' is now OFFLINE
DATASOURCE 'host3@alpha' IS NOW A MASTER
```

This does not recover the remaining slaves within the cluster, these must be manually recovered. This can be achieved either by using Section 4.5.1, "Recover a failed slave", or if this is not possible, using Section 4.5.1.1, "Provision or Reprovision a Slave".

### 4.5.2.2. Recover a shunned master

When a master datasource fails in `manual` policy mode, and the node has been failed over, once the datasource becomes available, the node can be added back to the dataservice by using the `recover` command, which enables the host as a slave:

```bash
[LOGICAL:EXPERT] /alpha > datasource host1 recover
VERIFYING THAT WE CAN CONNECT TO DATA SERVER 'host1'
DATA SERVER 'host1' IS NOW AVAILABLE FOR CONNECTIONS
RECOVERING 'host1@alpha' TO A SLAVE USING 'host2@alpha' AS THE MASTER
SETTING THE ROLE OF DATASOURCE 'host1@alpha' FROM 'master' TO 'slave'
RECOVERY OF 'host1@alpha' WAS SUCCESSFUL
```

The recovered master will added back to the dataservice as a slave.

### 4.5.2.3. Manually Failing over a Master in `maintenance` policy mode

If the dataservice is in `maintenance` mode when the master fails, automatic recovery cannot sensibly make the decision about which node should be used as the master. In that case, the dataservice service must be manually reconfigured.

In the sample below, `host1` is the current master, and `host2` is a slave. To manually update and switch `host1` to be the slave and `host2` to be the master:

1. Shun the failed master (`host1`) and set the replicator offline:

```bash
[LOGICAL:EXPERT] /alpha > datasource host1 shun
DataSource 'host1' set to SHUNNED
[LOGICAL:EXPERT] /alpha > replicator host1 offline
Replicator 'host1' is now OFFLINE
```

The failed master is shunned and the replicator is set offline, then the new master is set as follows:

```bash
[LOGICAL:EXPERT] /alpha > set master host2
VERIFYING THAT WE CAN CONNECT TO DATA SERVER 'host2'
DATA SERVER 'host2' IS NOW AVAILABLE FOR CONNECTIONS
DataSource 'host2' is now OFFLINE
DATASOURCE 'host2@alpha' IS NOW A MASTER
```

The new master is set, and the replicator is brought back online:)

```bash
[LOGICAL:EXPERT] /alpha > replicator host2 online
Replicator 'host2' is now ONLINE
```

The replicator is now active, and the new master is set.

The recovered master will added back to the dataservice as a slave.

```bash
[LOGICAL:EXPERT] /alpha > datasource host1 recover
VERIFYING THAT WE CAN CONNECT TO DATA SERVER 'host1'
DATA SERVER 'host1' IS NOW AVAILABLE FOR CONNECTIONS
RECOVERING 'host1@alpha' TO A SLAVE USING 'host2@alpha' AS THE MASTER
SETTING THE ROLE OF DATASOURCE 'host1@alpha' FROM 'master' TO 'slave'
RECOVERY OF 'host1@alpha' WAS SUCCESSFUL
```
2. Shun the slave host2 and set the replicator to the offline state:

```plaintext
[LOGICAL:EXPERT] /alpha > datasource host2 shun
DataSource 'host2' set to SHUNNED
[LOGICAL:EXPERT] /alpha > replicator host2 offline
Replicator 'host2' is now OFFLINE
```

3. Configure host2 as the master within the replicator service:

```plaintext
[LOGICAL:EXPERT] /alpha > replicator host2 master
```

4. Set the replicator on host2 online:

```plaintext
[LOGICAL:EXPERT] /alpha > replicator host2 online
```

5. Recover host2 online and then set it online:

```plaintext
[LOGICAL:EXPERT] /alpha > datasource host2 welcome
[LOGICAL:EXPERT] /alpha > datasource host2 online
```

6. Switch the replicator to be in slave mode:

```plaintext
[LOGICAL:EXPERT] /alpha > replicator host1 slave host2
Replicator 'host1' is now a slave of replicator 'host2'
```

7. Switch the replicator online:

```plaintext
[LOGICAL:EXPERT] /alpha > replicator host1 online
Replicator 'host1' is now ONLINE
```

8. Switch the datasource role for host1 to be in slave mode:

```plaintext
[LOGICAL:EXPERT] /alpha > datasource host1 slave
Datasource 'host1' now has role 'slave'
```

9. The configuration and roles for the host have been updated, the datasource can be added back to the dataservice and then put online:

```plaintext
[LOGICAL:EXPERT] /alpha > datasource host1 recover
DataSource 'host1' is now OFFLINE
[LOGICAL:EXPERT] /alpha > datasource host1 online
Setting server for data source 'host1' to READ-ONLY
```

```plaintext
+----------------------------------------------------------------------------+
|Variable_name  Value                                                        |
|read_only  ON                                                               |
+----------------------------------------------------------------------------+
DataSource 'host1@alpha' is now ONLINE
```

10. With the dataservice in automatic policy mode, the datasource will be placed online, which can be verified with `ls`:

```plaintext
[LOGICAL:EXPERT] /alpha > ls
COORDINATOR[host3:AUTOMATIC:ONLINE]

ROUTERS:
|connector@host1[19869](ONLINE, created=0, active=0) |
|connector@host2[28116](ONLINE, created=0, active=0) |
|connector@host3[1533](ONLINE, created=0, active=0) |

DATASOURCES:
|host1(slave:ONLINE, progress=156325, latency=725.737) |
|STATUS [OK] [2013/05/14 01:06:08 PM BST] |
|MANAGER(state=ONLINE) |
|REPLICATOR(role=slave, master=host2, state=ONLINE) |
|DATASERVER(state=ONLINE) |
|CONNECTIONS(created=0, active=0) |

|host2(master:ONLINE, progress=156325, TIL latency=0.606) |
|STATUS [OK] [2013/05/14 12:53:41 PM BST] |
|MANAGER(state=ONLINE) |
|REPLICATOR(role=master, state=ONLINE) |
|DATASERVER(state=ONLINE) |
```
4.5.2.4. Failing over a master

When a master datasource fails in *MANUAL* policy mode, the datasource must be manually failed over to an active datasource, either by selecting the most up to date slave automatically:

```
[LOGICAL:EXPERT] /alpha > failover
```

Or to an explicit host:

```
[LOGICAL:EXPERT] /alpha > failover to host2
```

For the *failover* command to work, the following conditions must be met:

- There must be a master or relay in the *SHUNNED* or *FAILED* state.
- There must be at least one slave in the *ONLINE* state.

If there is not already a *SHUNNED* or *FAILED* master and a failover must be forced, use *datasource shun* on the master, or failover to a specific slave.

4.6. Composite Cluster Switching, Failover and Recovery

Switching of a datasource is done to transfer the Master role from one cluster to another, usually in another datacenter site. This also has the effect of turning the original Master into to a Relay. The master datasource within a composite cluster can be forced to failover to the slave datasource in the event the master datasource is offline.

Switching the master datasource performs the following steps:

1. Set the master node to offline state. New connections to the master are rejected, and writes to the master are stopped.
2. On the relay in the target cluster, switch the datasource offline. New connections are rejected, stopping reads on this master.
3. Kill any outstanding client connections to the master data source, except those belonging to the *tungsten* account.
4. Send a heartbeat transaction between the old master and the new master, and wait until this transaction has been received. Once received, the THL on master and slave are up to date.
5. Perform the switch:
   - Configure all remaining replicators offline
   - Configure the target cluster relay node as the new master.
   - Set the new master to the online state.
   - New connections to the master are permitted.
6. Configure the old master to be a relay datasource.
7. Configure the slaves in the primary site to use the new master datasource.
8. Configure the slaves in the slave site to use the new relay datasource.
9. Update the connector configurations and enable client connections to connect to the masters and slaves.
The switching process is monitored by Continuent Tungsten, and if the process fails, either due to a timeout or a recoverable error occurs, the switch operation is rolled back, returning the dataservice to the original configuration. This ensures that the dataservice remains operational. In some circumstances, when performing a manual switch, the command may need to be repeated to ensure the requested switch operation completes.

The process takes a finite amount of time to complete, and the exact timing and duration will depend on the state, health, and database activity on the dataservice. The actual time taken will depend on how up to date the slave being promoted is compared to the master. The switch will take place regardless of the current status after a delay period.

### 4.6.1. Composite Cluster Site Switch

Our example cluster has two sites, east and west. They are both members of the composite cluster global. Site east has hosts db1, db2 and db3. Site west has hosts db4, db5 and db6.

```
$ ctrl -multi
Continuent Tungsten 2.0.3 build 3
east: session established
[LOGICAL] / > ls
+--------------------------------------------------------+
| DATA SERVICES:                                         |
+--------------------------------------------------------+
| east                                                   |
| west                                                   |
+--------------------------------------------------------+
$ use global
[LOGICAL] / > ls
COORDINATOR[db1:AUTOMATIC:ONLINE]
DATASOURCES:
| east[composite master:ONLINE]                          |
| STATUS [OK] [2015/04/14 01:26:26 AM UTC]               |
+--------------------------------------------------------+
| west[composite slave:ONLINE]                           |
| STATUS [OK] [2015/04/14 01:26:26 AM UTC]               |
+--------------------------------------------------------+

The detailed cluster status is shown below; click the icon to hide this detail:

Click the icon to show the detailed cluster status.

- **Composite Master Dataservice (Primary)** - east

```
[LOGICAL] /global > use east
[LOGICAL] /east > ls
COORDINATOR[db1:AUTOMATIC:ONLINE]
ROUTERS:
| connector@db1[9745](ONLINE, created=1, active=0) |
| connector@db2[9911](ONLINE, created=1, active=0) |
| connector@db3[9775](ONLINE, created=1, active=0) |
| connector@db4[9757](ONLINE, created=1, active=0) |
| connector@db5[9781](ONLINE, created=1, active=0) |
| connector@db6[9944](ONLINE, created=1, active=0) |
DATASOURCES:
| db1(master:ONLINE, progress=6, latency=0.814) |
| STATUS [OK] [2015/04/14 01:46:27 AM UTC] |
| MANAGER(state=ONLINE) |
| REPLICA(role=master, state=ONLINE) |
| DATASERVER(state=ONLINE) |
| CONNECTIONS(created=6, active=0) |
+--------------------------------------------------------+
| db2(slave:ONLINE, progress=6, latency=0.857) |
| STATUS [OK] [2015/04/14 01:46:27 AM UTC] |
| MANAGER(state=ONLINE) |
| REPLICA(role=slave, master=db1, state=ONLINE) |
| DATASERVER(state=ONLINE) |
+--------------------------------------------------------+
```
• Composite Slave Dataservice (DR) - west

  [LOGICAL] /east > use west
  [LOGICAL] /west > ls

  COORDINATOR:db4:AUTOMATIC:ONLINE

  ROUTERS:
  [connector@db1[9745](ONLINE, created=0, active=0)
  [connector@db2[9911](ONLINE, created=0, active=0)
  [connector@db3[9757](ONLINE, created=0, active=0)
  [connector@db4[9757](ONLINE, created=0, active=0)
  [connector@db5[9781](ONLINE, created=0, active=0)
  [connector@db6[9944](ONLINE, created=0, active=0)

  DATASOURCES:
  [db4(relay:ONLINE, progress=6, latency=5.050)
  [db5(slave:ONLINE, progress=6, latency=5.522)
  [db6(slave:ONLINE, progress=6, latency=5.501)

  Manually switch the composite master role to the other site:

  [LOGICAL] /global > ls
  COORDINATOR:db1:AUTOMATIC:ONLINE

  DATASOURCES:
  [east(composite slave:ONLINE)

  [LOGICAL] /global > switch
  SELECTED SLAVE: 'west@global'
  FLUSHING TRANSACTIONS THROUGH 'db1@east'
  REPLICATOR 'db1' IS NOW USING MASTER CONNECT URI 'thl://db4:2112/'
  composite data source 'west@global' is now OFFLINE
  PUT THE NEW MASTER 'west@global' ONLINE
  PUT THE PRIOR MASTER 'east@global' ONLINE AS A SLAVE
  REVERT POLICY: MAINTENANCE => AUTOMATIC
  SWITCH TO 'west@global' WAS SUCCESSFUL

  [LOGICAL] /global > ls
  COORDINATOR:db1:AUTOMATIC:ONLINE

  DATASOURCES:
  [east(composite slave:ONLINE)
  [status [OK] [2015/04/14 01:45:48 AM UTC]
The detailed cluster status is shown below; click the icon to hide this detail:

Click the icon to show the detailed cluster status.

- **Composite Slave Dataservice (DR) - east**

  ```
  [LOGICAL] /global > use east
  [LOGICAL] /east > ls
  COORDINATOR[db1:AUTOMATIC:ONLINE]
  
  ROUTERS:
  +----------------------------------------------------------------------------+
  | connector@db1[9745](ONLINE, created=1, active=0)              |
  | connector@db2[9911](ONLINE, created=1, active=0)              |
  | connector@db3[9775](ONLINE, created=1, active=0)              |
  | connector@db4[9757](ONLINE, created=1, active=0)              |
  | connector@db5[9781](ONLINE, created=1, active=0)              |
  | connector@db6[9944](ONLINE, created=1, active=0)              |
  +----------------------------------------------------------------------------+
  
  DATASOURCES:
  +----------------------------------------------------------------------------+
  | db1(relay:ONLINE, progress=3, latency=4.000)                                |
  | STATUS [OK] [2015/04/14 01:45:47 AM UTC]                                    |
  |  MANAGER(state=ONLINE)                                                     |
  |  REPLICA(role=relay, master=db4, state=ONLINE)                           |
  |  DASERVER(state=ONLINE)                                                   |
  |  CONNECTIONS(created=6, active=0)                                          |
  +----------------------------------------------------------------------------+
  
  +----------------------------------------------------------------------------+
  | db2(slave:ONLINE, progress=3, latency=5.188)                                |
  | STATUS [OK] [2015/04/14 01:45:48 AM UTC]                                    |
  |  MANAGER(state=ONLINE)                                                     |
  |  REPLICA(role=slave, master=db1, state=ONLINE)                            |
  |  DASERVER(state=ONLINE)                                                   |
  |  CONNECTIONS(created=0, active=0)                                          |
  +----------------------------------------------------------------------------+
  
  +----------------------------------------------------------------------------+
  | db3(slave:ONLINE, progress=3, latency=5.249)                                |
  | STATUS [OK] [2015/04/14 01:45:48 AM UTC]                                    |
  |  MANAGER(state=ONLINE)                                                     |
  |  REPLICA(role=slave, master=db1, state=ONLINE)                            |
  |  DASERVER(state=ONLINE)                                                   |
  |  CONNECTIONS(created=0, active=0)                                          |
  +----------------------------------------------------------------------------+
  ```

- **Composite Master Dataservice (Primary) - west**

  ```
  [LOGICAL] /east > use west
  west: session established
  [LOGICAL] /west > ls
  COORDINATOR[db4:AUTOMATIC:ONLINE]
  
  ROUTERS:
  +----------------------------------------------------------------------------+
  | connector@db1[9745](ONLINE, created=0, active=0)              |
  | connector@db2[9911](ONLINE, created=0, active=0)              |
  | connector@db3[9775](ONLINE, created=0, active=0)              |
  | connector@db4[9757](ONLINE, created=0, active=0)              |
  | connector@db5[9781](ONLINE, created=0, active=0)              |
  | connector@db6[9944](ONLINE, created=0, active=0)              |
  +----------------------------------------------------------------------------+
  
  DATASOURCES:
  +----------------------------------------------------------------------------+
  | db4(master:ONLINE, progress=3, THL latency=0.671)                                |
  | STATUS [OK] [2015/04/14 01:45:42 AM UTC]                                    |
  |  MANAGER(state=ONLINE)                                                     |
  |  REPLICA(role=master, state=ONLINE)                                        |
  +----------------------------------------------------------------------------+
  ```
4.6.2. Composite Cluster Site Failover (Forced Switch)

In the event the Primary site goes down, and a graceful manual switch is not possible, the composite master role can be failed over to the Disaster Recovery cluster using `cctrl`. The `failover` command performs the forced switch operation, annotating the progress.

In this example, hosts db1 (the composite master), db2 and db3 in cluster east have been shut down. To force dataservice `west` to become the primary, login to a node in that cluster and get into `cctrl`:

```
shell> cctrl -multi
Continuent Tungsten 2.0.5 build 3
west: session established
[LOGICAL] / > use global
[LOGICAL] /global > ls
COORDINATOR[db4:AUTOMATIC:ONLINE]
DATASOURCES:
|east(composite master:SHUNNED(FAILSAFE_SHUN))                               |
|STATUS [SHUNNED] [2015/04/14 01:46:59 AM UTC]                               |
+----------------------------------------------------------------------------+
|west(composite slave:ONLINE)                                                |
|STATUS [OK] [2015/04/14 01:46:59 AM UTC]                                    |
+----------------------------------------------------------------------------+
[LOGICAL] /global > failover
WARNING: DATA SERVICE 'east' IS NOT AVAILABLE. CANNOT GET STATE.
WARNING: CAN'T GET POLICY MODE FOR SERVICE 'east'. CONTINUING.
WARNING: CAN'T SET POLICY MODE 'maintenance' FOR SERVICE 'east'. CONTINUING.
SELECTED SLAVE: 'west@global'
WARNING: UNABLE TO REACH PHYSICAL DATA SERVICE 'east' AT THIS TIME.
EXCEPTION: Unable unauible to continue with command because no manager is available in service 'east'.

CONTINUING WITH COMMAND
ENSURING THAT WE CATCH UP WITH THE MOST ADVANCED RELAY
composite data source 'west@global' is now OFFLINE
WARNING: UNABLE TO REACH PHYSICAL DATA SERVICE 'west' AT THIS TIME.
EXCEPTION: Unable to unable to continue with command because no manager is available in service 'east'.

CONTINUING WITH COMMAND
PUT THE NEW MASTER 'west@global' ONLINE
WARNING: CAN'T SET POLICY MODE 'AUTOMATIC' FOR SERVICE 'west'. CONTINUING.
REVERT POLICY: MAINTENANCE => AUTOMATIC
FAILOVER TO 'west@global' WAS SUCCESSFUL
[LOGICAL] /global > ls
COORDINATOR[db4:AUTOMATIC:ONLINE]
DATASOURCES:
|east(composite master:SHUNNED(MANUAL-FAILOVER))                             |
|STATUS [SHUNNED] [2015/04/14 02:13:18 AM UTC]                               |
+----------------------------------------------------------------------------+
|west(composite master:ONLINE)                                               |
```
The detailed cluster status is shown below; click the icon to hide this detail:

Click the icon to show the detailed cluster status.

- **Composite Master Dataservice (Primary)** - *west*

  ```
  [LOGICAL] /global > use west
  [LOGICAL] /west > ls
  COORDINATOR[db4:AUTOMATIC:ONLINE]
  ROUTERS:
  [connector@db4[9757] (ONLINE, created=0, active=0)]
  [connector@db5[9781] (ONLINE, created=0, active=0)]
  [connector@db6[9944] (ONLINE, created=0, active=0)]
  DATASOURCES:
  [db4(master:ONLINE, progress=7, THL latency=0.110)]
  [db5(slave:ONLINE, progress=7, latency=0.172)]
  [db6(slave:ONLINE, progress=7, latency=0.173)]
  ```

### 4.6.3. Composite Cluster Site Recovery

The first step in recovering the SHUNNED dataservice is to re-provision the nodes if the data has gotten out of sync. See Section 4.5.1.1, “Provision or Reprovision a Slave” for more information.

Once the failed site has been restored, the shunned/superseded dataservice can be brought back online using `cctrl`. The `recover` command performs this operation, annotating the progress.

```shell
$ cctrl -multi
Continuent Tungsten 2.0.5 build 3
west: session established
[LOGICAL] / > use global
[LOGICAL] /global > ls
COORDINATOR[db4:AUTOMATIC:ONLINE]
DATASOURCES:
[db4(composite master:SHUNNED(SUPERSEDED))]
[db5(slave:ONLINE, progress=7, latency=0.172)]
[db6(slave:ONLINE, progress=7, latency=0.173)]
```

The detailed cluster status is shown below; click the icon to hide this detail:
Click the icon to show the detailed cluster status.

- **SHUNNED(SUPERSEDED) Composite Master Dataservice - east**

  ```
  [LOGICAL] / > use east
  [LOGICAL] /east > ls
  COORDINATOR[db2:AUTOMATIC:ONLINE]
  ROUTERS:
  
  | connector@db1[10051] (ONLINE, created=0, active=0) |
  | connector@db2[16111] (ONLINE, created=0, active=0) |
  | connector@db3[16036] (ONLINE, created=0, active=0) |
  | connector@db4[9757] (ONLINE, created=1, active=0) |
  | connector@db5[9781] (ONLINE, created=1, active=0) |
  | connector@db6[9944] (ONLINE, created=1, active=0) |
  
  DATASOURCES:
  
  | db1(master:SHUNNED(SUPERSEDED), progress=7, THL latency=0.934) |
  | STATUS [SHUNNED] [2015/04/14 02:28:53 AM UTC] |
  |   MANAGER(state=ONLINE) |
  |   REPLICATOR(role=master, state=ONLINE) |
  |   DATASERVER(state=ONLINE) |
  |   CONNECTIONS(created=3, active=0) |
  
  | db2(slave:SHUNNED(SUPERSEDED), progress=7, latency=6.488) |
  | STATUS [SHUNNED] [2015/04/14 02:28:53 AM UTC] |
  |   MANAGER(state=ONLINE) |
  |   REPLICATOR(role=slave, master=db1, state=ONLINE) |
  |   DATASERVER(state=ONLINE) |
  |   CONNECTIONS(created=0, active=0) |
  
  | db3(slave:SHUNNED(SUPERSEDED), progress=7, latency=11.164) |
  | STATUS [SHUNNED] [2015/04/14 02:28:53 AM UTC] |
  |   MANAGER(state=ONLINE) |
  |   REPLICATOR(role=slave, master=db1, state=ONLINE) |
  |   DATASERVER(state=ONLINE) |
  |   CONNECTIONS(created=0, active=0) |
  ```

Use the `recover` to bring the SHUNNED dataservice back online as a composite slave:

```
[LOGICAL] /global > recover
IDENTIFIED DATASOURCE 'east@global' FOR RECOVERY
COULD NOT IDENTIFY ACTIVE PRIMARY FOR SERVICE 'east'
ATTEMPTING TO IDENTIFY A FAILED PRIMARY FOR 'east'
PHYSICAL DATA SERVICE 'east' DOES NOT HAVE AN ACTIVE RELAY
FORCING THE PHYSICAL RELAY TO BE 'db1'
DATASOURCE 'db1east' IS NOW A RELAY
RECOVERED 2 DATA SOURCES IN SERVICE 'east'
composite data source 'east@global' role is now SLAVE
composite data source 'east' is now OFFLINE
REVERT SET POLICY AUTOMATIC
RECOVERY OF COMPOSITE SERVICE 'global' IS COMPLETE
[LOGICAL] /global > ls
COORDINATOR[db2:AUTOMATIC:ONLINE]
DATASOURCES:

| east[composite slave:ONLINE] |
| STATUS [OK] [2015/04/14 04:12:01 AM UTC] |

| west[composite master:ONLINE] |
| STATUS [OK] [2015/04/14 02:28:53 AM UTC] |
```

The detailed cluster status is shown below; click the icon to hide this detail:

Click the icon to show the detailed cluster status.
4.7. 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

4.7.1. 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.

Within `cctrl` the status of the datasource will be marked as `DIMINISHED`, and the replicator state as `SUSPECT`.

LOGICAL] /alpha > ls
More detailed information about the status and the statement that failed can be obtained within `cctrl` using the `replicator` command:

```
[LOGICAL] /alpha > replicator host2 status
```

The `trepsvc.log` log file will also contain the error information about the failed statement. For example:

```
INFO | jvm 1 | 2013/06/26 10:14:12 | 2013-06-26 10:14:12,423 [firstcluster - »
  q-to-dbms-0] INFO pipeline.SingleThreadStageTask Performing emergency »
  rollback of applied changes
INFO | jvm 1 | 2013/06/26 10:14:12 | 2013-06-26 10:14:12,424 [firstcluster - »
  q-to-dbms-0] INFO pipeline.SingleThreadStageTask Dispatching error event: »
  Event application failed: seqno=120 fragment=0 message=java.sql.SQLException: Statement failed on slave but succeeded on master
INFO | jvm 1 | 2013/06/26 10:14:12 | 2013-06-26 10:14:12,424 [firstcluster - »
  q-to-dbms-0] INFO pipeline.SingleThreadStageTask Performing emergency »
  rollback of applied changes
INFO | jvm 1 | 2013/06/26 10:14:12 | 2013-06-26 10:14:12,425 [firstcluster - »
  q-to-dbms-0] INFO pipeline.SingleThreadStageTask Dispatching error event: »
  Event application failed: seqno=120 fragment=0 message=java.sql.SQLException: Statement failed on slave but succeeded on master
Once the error or problem has been found, the exact nature of the error should be determined so that a resolution can be identified:

1. Identify the reason for the failure by examining the full error message. Common causes are:

   - **Duplicate primary key**
     
     A row or statement is being inserted or updated that already has the same insert ID or would generate the same insert ID for tables that have auto increment enabled. The insert ID can be identified from the output of the transaction using `thl`. Check the 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 database.

     The error will normally be identified due to the following error message when viewing the current replicator status, for example:

     ```
     [LOGICAL] /alpha > replicator host3 status
     ...
     pendingError       : Event application failed: seqno=10 fragno=0 »
     message:java.sql.SQLException: Statement failed on slave but succeeded on master
     pendingErrorCode   : NONE
     pendingErrorEventId: mysql-bin.000032:0000000000001872;0
     pendingErrorSeqno  : 10
     pendingExceptionMessage: java.sql.SQLException: Statement failed on slave but succeeded on master
     "insert into myent values (0,"Test Message")"
     ...
     ```

     The error can be generated when an insert or update has taken place on the 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:

     ```
     shell> thl list -seqno 10
     SEQ# = 10 / FRAG# = 0 (last frag)
     - TIME = 2014-01-09 16:47:40.0
     - EPOCH# = 1
     - EVENTID = mysql-bin.000032:0000000000001872;0
     - SOURCEID = host1
     - METADATA = [mysql_server_id=1;dbms_type=mysql;service=firstcluster;shard=test]
     - TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
     - SQL(0) = SET INSERT_ID = 2
     - OPTIONS = [##charset = UTF-8, autocommit = 1, sql_auto_is_null = 0, foreign_key_checks = 1, unique_checks = 1, sql_mode = '', character_set_client = 33, collation_connection = 33, collation_server = 8]
     - SCHEMA = test
     - SQL(1) = insert into myent values (0,"Test Message")
     ```

     In this example, an `INSERT` operation is inserting a new row. The generated insert ID is also shown (in line 9, `SQL(0)`). Check the destination database and determine the what the current value of the corresponding row:

     ```
     mysql> select * from myent where id = 2;
     +----+---------------+
     | id | msg           |
     +----+---------------+
     |  2 | Other Message |
     +----+---------------+
     1 row in set (0.00 sec)
     ```

     The actual row values are different, which means that either value may be correct. In complex data structures, there may be multiple statements or rows that trigger this error if following data also relies on this value.

     For example, if multiple rows have been inserted on the 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 4.7.2, “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:

  ```
  [LOGICAL] /alpha > replicator host3 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.

  To perform a backup and restore, see Section 4.9, “Creating a Backup”, or Section 4.10, “Restoring a Backup”.

### 4.7.2. 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 4.7.1, “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.

4.8. Deploying Automatic Replicator Recovery

Automatic recovery enables the replicator to go back ONLINE in the event of a transient failure that is triggered during either the ONLINE or GOING-ONLINE SYNCHRONIZING state that would otherwise trigger a change of states to OFFLINE. For example, connection failures, or restarts in the MySQL service, trigger the replicator to go OFFLINE. With autorecovery enabled, the replicator will attempt to put the replicator ONLINE 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 within the configurable reset interval, then the replicator will go to the OFFLINE state.

3. If the replicator remains in the ONLINE 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` Set 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` 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` 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` 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.

4.9. Creating a Backup

The `backup` command for a datasource within `cctrl` 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.

Backings up a datasource can occur while the replicator is online:

```
[LOGICAL:EXPERT] /alpha > datasource host3 backup
Using the 'mysqldump' backup agent.
Replicator 'host3' starting 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   4096 Apr  4 16:08 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 E.1.1, “The backups Directory”.

The `storage.index` contains the backup file index information. The actual backup data is stored in the GZipped file. The properties of the backup file, including the tool used to create the backup, and the checksum information, are location in the corresponding `.properties` file. Note that each backup and property file is uniquely numbered so that it can be identified when restoring a specific backup.

A backup can also be initiated and run in the background by adding the & (ampersand) to the command:

```
[LOGICAL:EXPERT] /alpha > datasource host3 backup &
[1] datasource host3 backup - RUNNING
YOU MUST BE USING A DATA SERVICE TO EXECUTE THIS COMMAND
EXECUTE 'use <data service name>' TO SET YOUR CONTEXT.
[1] datasource host3 backup - SUCCESS
```

### 4.9.1. Using a Different Backup Tool

If `xtrabackup` is installed when the dataservice is first created, `xtrabackup` will be used as the default backup method. Four built-in backup methods are provided:

- **mysqldump** — SQL dump to a single file. This is the easiest backup method but it is not appropriate for large data sets.
- **xtrabackup** — Full backup to a GZipped file. This will save space but it takes longer to take the backup and to restore.
- **xtrabackup-full** — Full backup to a directory (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 the this tool is optional and can configured during installation, or afterwards by updating the configuration using `tpm`.

To update the configuration to use `xtrabackup`, install the tool and then follow the directions for `tpm update` to apply the `--repl-backup-method=xtrabackup-full` setting.

To use `xtrabackup-full` without changing the configuration, specify the backup agent to the `backup` command within `cctrl`:

```
[LOGICAL:EXPERT] /alpha > datasource host2 backup xtrabackup-full
Replicator 'host2' starting backup
Backup of dataSource 'host2' succeeded; uri=storage://file-system/store-0000000006.properties
```

### 4.9.2. Automating Backups

Backups cannot be automated within Continuent Tungsten, instead a `cron` job should be used to automate the backup process. `cluster_backup` is packaged with Continuent Tungsten to provide a convenient interface with `cron`. The `cron` entry should be added to every datasource or active witness in the cluster. The command includes logic to ensure that it will only take one backup per cluster by only running on the current coordinator. See Section 7.6, “The cluster_backup Command” for more information.

```
shell> /opt/continuent/tungsten/cluster-home/bin/cluster_backup.sh >
```


The command output will be stored in `/opt/continuent/service_logs/cluster_backup.log` for later review. Use your preferred mechanism to configure cron to execute this command on the desired schedule. Alternatively, you can call the backup command directly through `cctrl`. This method does not ensure the named datasource is `ONLINE` or even available to be backed up.

```shell
shell> echo "datasource host2 backup" | /opt/continuent/tungsten/tungsten-manager/bin/cctrl -expert
```

### 4.9.3. Using a Different Directory Location

The default backup location is the `backups` directory of the Continuent Tungsten installation directory. For example, using the recommended installation location, backups are stored in `/opt/continuent/backups`.

See Section E.1.1.4, “Relocating Backup Storage” for details on changing the location where backups are stored.

### 4.9.4. Creating an External Backup

There are several considerations to take into account when you are using a tool other than Continuent Tungsten 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 4.10.2, “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
shell> mysqldump --opt --single-transaction --all-databases --add-drop-database --master-data=2
```

### 4.10. 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 4.5.1.1, “Provision or Reprovision a Slave”.

To restore a backup, use the `restore` command to a datasource within `cctrl`:

1. Shun the datasource to be restored, and put the replicator service offline using `cctrl`:

   ```shell
   [LOGICAL] /alpha > datasource host2 shun
   [LOGICAL] /alpha > replicator host2 offline
   ```

2. Restore the backup using `cctrl`:

   ```shell
   [LOGICAL] /alpha > datasource host2 restore
   ```

   By default, the restore process takes the latest backup available for the host being restored. Continuent Tungsten does not automatically locate the latest backup within the dataservice across all datasources.

#### 4.10.1. Restoring a Specific Backup

To restore a specific backup, specify the location of the corresponding properties file using the format:
For example, to restore the backup from the filesystem using the information in the properties file `store-0000000004.properties`, login to the failed host:

1. Shun the datasource to be restored, and put the replicator service offline using `cctrl`:

   ```
   [LOGICAL] /alpha > datasource host2 shun
   [LOGICAL] /alpha > replicator host2 offline
   ```

2. Restore the backup using `cctrl`:

   ```
   [LOGICAL] /alpha > datasource host2 restore storage://file-system/store-0000000004.properties
   ```

   The supplied location is identical to that returned when a backup is performed.

### 4.10.2. Restoring an External Backup

If a backup has been performed outside of Continent Tungsten, for example from filesystem snapshot or a backup performed outside of the dataservice, follow these steps:

1. Shun the datasource to be restored, and put the replicator service offline using `cctrl`:

   ```
   [LOGICAL:EXPERT] /alpha > datasource host2 shun
   [LOGICAL:EXPERT] /alpha > replicator host2 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 4.9.4, "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
     ```

   - 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. Recover the datasource using `cctrl`:

   ```
   [LOGICAL] /alpha > datasource host2 recover
   ```

   The `recover` command will start the dataserver if it was left running and then bring the replicator and other operations online.

### 4.10.3. 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 4.5.1.1, "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:

```
[LOGICAL:EXPERT] /alpha > datasource host2 backup
Using the 'xtrabackup' backup agent. 
Replicator 'host2' starting backup 
Backup of dataSource 'host2' succeeded; uri=storage://file-system/store-0000000006.properties
```

2. Copy the backup information from host2 to host3. See Section E.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-8120968634645699665.sql host3:$PWD/
store-0000000006.properties
```

3. Shun the datasource to be restored, and put the replicator service offline using cctrl:

```
[LOGICAL] /alpha > datasource host2 shun
[LOGICAL] /alpha > replicator host2 offline
```

4. Restore the backup using cctrl:

```
[LOGICAL] /alpha > datasource host2 restore
```

Once the restore operation has completed, the datasource will be placed into the online state.

---

**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.

### 4.10.4. 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/mysql.error.log:

```
130520 14:37:08 [Note] Recovering after a crash using /var/log/mysql/mysql-bin
130520 14:37:08 [Note] Starting crash recovery...
130520 14:37:08 [Note] Crash recovery finished.
130520 14:37:08 [Note] Server hostname (bind-address): '0.0.0.0'; port: 13306
130520 14:37:08 [Note] - '0.0.0.0' resolves to '0.0.0.0';
130520 14:37:08 [Note] Server socket created on IP: '0.0.0.0'.
130520 14:37:08 [ERROR] Fatal error: Can't open and lock privilege tables: Table 'mysql.host' doesn't exist
130520 14:37:08 [ERROR] /usr/sbin/mysqld: File '/var/run/mysqld/mysqld.pid' not found (Errcode: 13)
130520 14:37:08 [ERROR] /usr/sbin/mysqld: Error reading file 'UNKNOWN' (Errcode: 9)
130520 14:37:08 [ERROR] /usr/sbin/mysqld: Error on close of 'UNKNOWN' (Errcode: 9)
```

Performing a restore operation on this 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. Shun the host2 datasource to be restored, and put the replicator service offline using cctrl:

```
[LOGICAL] /alpha > datasource host2 shun
[LOGICAL] /alpha > replicator host2 offline
```

2. Shun the host3 datasource to be restored, and put the replicator service offline using cctrl:

```
[LOGICAL] /alpha > datasource host3 shun
[LOGICAL] /alpha > replicator host3 offline
```

3. Stop the mysqld service on host2:

```
shell> sudo /etc/init.d/mysqld stop
```

4. Stop the mysqld service on host3:
5. Delete the *mysqld* data directory on host2:

```shell
shell> sudo rm -rf /var/lib/mysql/*
```

6. If necessary, ensure the *tungsten* user can write to the MySQL directory:

```shell
shell> sudo chmod 777 /var/lib/mysql
```

7. Use *rsync* on host3 to send the data files for MySQL to host2:

```shell
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. Recover host3 back to the dataservice:

```shell
[LOGICAL:EXPERT] /alpha > datasource host3 recover
```

9. Update the ownership and permissions on the data files on host2:

```shell
host2 shell> sudo chown -R mysql:mysql /var/lib/mysql
host2 shell> sudo chmod 770 /var/lib/mysql
```

10. Recover host2 back to the dataservice:

```shell
[LOGICAL:EXPERT] /alpha > datasource host2 recover
```

The *recover* command will start MySQL and ensure that the server is accessible before restarting replication. If the MySQL instance does not start; correct any issues and attempt the *recover* command again.

### 4.10.5. Rebuilding a Lost Datasource

If a datasource has been lost within the dataservice, for example, a complete hardware failure or disk crash, the datasource can be added back to the cluster once the operating system and other configuration have been completed. Essentially, the process is the same as when initially setting up your node, with the node being re-confirmed as part of the running service, installing and configuring only the returning node to the cluster.

In the following steps, the host host3 is being recovered into the cluster:

1. Setup the host with the prerequisites, as described in Appendix C, Prerequisites.

2. Restore a snapshot of the data taken from another slave into the dataserver. If you have existing backups of this slave or another, they should be used. If not, take a snapshot of an existing slave and use this to apply the data to the slave. This will need to be performed outside of the Continuent Tungsten service using the native restore method for the backup method you have chosen. The backup must include the entire schema of your database, including the *tungsten* schemas for your services.

3. The next steps depend on the availability of the hostname. If the hostname of the datasource that was lost can be reused, then the host can be reconfigured within the existing service. If the hostname is not available, the service must be reconfigured to remove the old host, and add the new host.

**Reusing an Existing Hostname**

a. Login in to the server used for staging your Continuent Tungsten installation, and change to the staging directory. To determine the staging directory, use:

```shell
shell> tpm query staging
```

b. Repeat the installation of the service on the host being brought back:

```shell
shell> /tools/tpm update svc_name --hosts=host3
```

The update process will re-install Continuent Tungsten on the host specified without reacting to the existence of the *tungsten* schema in the database.

**Removing and Adding a New Host**
a. Remove the existing (lost) datasource from the cluster using `cctrl`. First switch to administrative mode:

```
[LOGICAL] /alpha > admin
```

Remove the host from the dataservice:

```
[ADMIN] /alpha > rm host3
```

WARNING: This is an expert-level command:
Incorrect use may cause data corruption
or make the cluster unavailable.

Do you want to continue? (y/n)>

b. Login in to the server used for staging your Continuent Tungsten installation, and change to the staging directory. To determine the staging directory, use:

```
shell> tpm query staging
```

c. Update the dataservice configuration with the new datasource, the example below uses `host4` as the replacement datasource. The `--dataservice-master-host` should be used to specify the current master in the cluster:

```
shell> /tools/tpm configure svc_name --dataservice-hosts=host1,host2,host4  
     --dataservice-connectors=host1,host2,host4  
     --dataservice-master-host=host4
```

d. Update the installation across all the hosts:

```
shell> /tools/tpm update svc_name
```

4. Use `cctrl` to check and confirm the operation of the restore datasource.

The restored host should be part of the cluster and accepting events from the master as configured.

### 4.10.6. Resetting an Entire Dataservice from Filesystem Snapshots

To restore an entire dataservice from filesystem snapshots, the steps below should be followed. The same snapshot should be used on each host so that data on each host is the same. The following steps should be followed:

1. Set the dataservice into the `MAINTENANCE` policy mode:

```
[LOGICAL:EXPERT] /alpha > set policy maintenance
```

2. The following steps must be completed on each server before completing the next step:

   a. Stop the Continuent Tungsten services:

```
shell> stopall
```

   b. Stop MySQL:

```
shell> sudo /etc/init.d/mysql stop
```

   c. Replace the MySQL data files with the filesystem or snapshot data.

   d. Delete the THL files for each of the services that need to be reset:

```
shell> rm /opt/continuent/thl/*
```

   e. Start MySQL to perform maintenance on the Tungsten schemas:

```
shell> sudo /etc/init.d/mysql start
```

   f. Delete any Tungsten service schemas:

```
mysql> DROP DATABASE tungsten_alpha;
```

Once these steps have been executed on all the servers in the cluster, the services can be restarted.

3. On the current master, start the Continuent Tungsten services:

```
shell> startall
```

Now start the services using the same command on each of the remaining servers.
4.11. Performing Database or OS Maintenance

When performing database or operating system maintenance, datasources should be temporarily removed from the dataservice using the `datasource shun` command. 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.

4.11.1. Performing Maintenance on a Single Slave

Performing maintenance on a single slave can be achieved by temporarily shunning the slave (while in AUTOMATIC policy mode) and doing the necessary maintenance. Shunning a datasource in this way will temporarily remove it from the dataservice, and prevent active and new connections from using the datasource for operations.

The steps are:

1. Shun the slave:
   ```
   [LOGICAL:EXPERT] /alpha > datasource host2 shun
   ```
   Shunning a datasource does not put the replicator offline, so the replicator should also be put in the offline state to prevent replication and changes being applied to the database:
   ```
   [LOGICAL:EXPERT] /alpha > replicator host2 offline
   ```

2. Perform the required maintenance, including updating the operating system, software or hardware changes.

3. Validate the server configuration:
   ```
   shell> tpm validate
   ```

4. Recover the slave back to the dataservice:
   ```
   [LOGICAL:EXPERT] /alpha > datasource host2 recover
   ```
   Once the datasource is added back to the dataservice, the status of the node should be checked to ensure that the datasource has been correctly added back, and the node is ONLINE and up to date.

While the datasource is shunned, the node can be shutdown, restarted, upgraded, or any other maintenance. Throughout the process, the slave should be monitored to ensure that the datasource is correctly added back into the dataservice, and has caught up with the master. Any problems should be addressed immediately.

4.11.2. Performing Maintenance on a Master

Master maintenance must be carried out when the master has been switched to a slave, and then shunned. The master can be temporarily switched to a slave, taken out of the dataservice through shunning, and then added back to the dataservice and then switched back again to be the master.

```
Important

Maintenance on the dataserver should be performed directly on the corresponding server, not through the connector.
```

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>Set the maintenance policy</td>
<td>set policy maintenance</td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>3</td>
<td>Switch master</td>
<td>switch to host2</td>
<td>Slave</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>4</td>
<td>Shun host1</td>
<td>datasource host1 shun</td>
<td>Shunned</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>5</td>
<td>Perform maintenance</td>
<td></td>
<td>Shunned</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>6</td>
<td>Validate the host1 server</td>
<td>tpm validate</td>
<td>Shunned</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>7</td>
<td>Recover the slave (host1)</td>
<td>datasource host1 recover</td>
<td>Slave</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>8</td>
<td>Ensure the slave has caught up</td>
<td></td>
<td>Slave</td>
<td>Master</td>
<td>Slave</td>
</tr>
</tbody>
</table>
### 4.11.3. Performing Maintenance on an Entire Dataservice

To perform maintenance on all of the machines within a dataservice, 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 maintenance policy</td>
<td>set policy maintenance</td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>3</td>
<td>Shun slave <strong>host2</strong></td>
<td>datasource <strong>host2</strong> shun</td>
<td>Master</td>
<td>Slave</td>
<td>Shunned</td>
</tr>
<tr>
<td>4</td>
<td>Perform maintenance</td>
<td></td>
<td>Master</td>
<td>Slave</td>
<td>Shunned</td>
</tr>
<tr>
<td>5</td>
<td>Validate the <strong>host2</strong> server configuration</td>
<td>tpm validate</td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>6</td>
<td>Recover the slave <strong>host2</strong> back</td>
<td>datasource <strong>host2</strong> recover</td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>7</td>
<td>Ensure the slave (<strong>host2</strong>) has caught up</td>
<td></td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>8</td>
<td>Shun slave <strong>host3</strong></td>
<td>datasource <strong>host3</strong> shun</td>
<td>Master</td>
<td>Slave</td>
<td>Shunned</td>
</tr>
<tr>
<td>9</td>
<td>Perform maintenance</td>
<td></td>
<td>Master</td>
<td>Slave</td>
<td>Shunned</td>
</tr>
<tr>
<td>10</td>
<td>Validate the <strong>host3</strong> server configuration</td>
<td>tpm validate</td>
<td>Master</td>
<td>Slave</td>
<td>Shunned</td>
</tr>
<tr>
<td>11</td>
<td>Recover slave <strong>host3</strong> back</td>
<td>datasource <strong>host3</strong> recover</td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>12</td>
<td>Ensure the slave (<strong>host3</strong>) has caught up</td>
<td></td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>13</td>
<td>Switch master to <strong>host2</strong></td>
<td>switch to host2</td>
<td>Slave</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>14</td>
<td>Shun <strong>host1</strong></td>
<td>datasource <strong>host1</strong> shun</td>
<td>Shunned</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>15</td>
<td>Perform maintenance</td>
<td></td>
<td>Shunned</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>16</td>
<td>Validate the <strong>host1</strong> server configuration</td>
<td>tpm validate</td>
<td>Shunned</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>17</td>
<td>Recover the slave <strong>host1</strong> back</td>
<td>datasource <strong>host1</strong> recover</td>
<td>Slave</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>18</td>
<td>Ensure the slave (<strong>host1</strong>) has caught up</td>
<td></td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>19</td>
<td>Switch master back to <strong>host1</strong></td>
<td>switch to host1</td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>20</td>
<td>Set automatic policy</td>
<td>set policy automatic</td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
</tbody>
</table>

### 4.11.4. Making Online Schema Changes

Similar to the maintenance procedure, schema changes to an underlying dataserver may need to be performed on dataservers that are not part of an active dataservice. Although many inline schema changes, such as the addition, removal or modification of an existing table definition will be correctly replicated to slaves, other operations, such as creating new indexes, or migrating table data between table definitions, is best performed individually on each dataserver while it has been temporarily taken out of the dataservice.

The basic process is to temporarily shun each slave, perform the schema update, and then recover the slave back to the dataservice.
Operations supported by these online schema changes must be backwards compatible. Changes to the schema on slaves that would otherwise break the replication cannot be performed using the online method.

**Important**

While a slave is in the SHUNNED state, Continuent Tungsten will have switched the server to read-only mode. You must use a user with SUPER privileges to execute the schema change statements to bypass this read-only restriction.

The following method assumes a schema update on the entire dataservice by modifying the schema on the slaves first. The schema shows three datasources being updated in sequence, slaves first, then the master.

<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>Shun slave host2</td>
<td>datasource host2 shun</td>
<td>Master</td>
<td>Shunned</td>
<td>Slave</td>
</tr>
<tr>
<td>3</td>
<td>Connect to dataserver on host2 and update schema</td>
<td></td>
<td>Master</td>
<td>Shunned</td>
<td>Slave</td>
</tr>
<tr>
<td>4</td>
<td>Recover slave back</td>
<td>datasource host2 recover</td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>5</td>
<td>Ensure the slave (host2) has caught up</td>
<td>ls</td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>6</td>
<td>Shun slave host3</td>
<td>datasource host3 shun</td>
<td>Master</td>
<td>Slave</td>
<td>Shunned</td>
</tr>
<tr>
<td>7</td>
<td>Connect to dataserver on host3 and update schema</td>
<td></td>
<td>Master</td>
<td>Slave</td>
<td>Shunned</td>
</tr>
<tr>
<td>8</td>
<td>Recover slave back</td>
<td>datasource host3 recover</td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>9</td>
<td>Ensure the slave (host3) has caught up</td>
<td>ls</td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>10</td>
<td>Switch master to host2</td>
<td>switch to host2</td>
<td>Slave</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>11</td>
<td>Shun host1</td>
<td>datasource host1 shun</td>
<td>Shunned</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>12</td>
<td>Connect to dataserver on host1 and update schema</td>
<td></td>
<td>Shunned</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>13</td>
<td>Recover host1 back</td>
<td>datasource host1 recover</td>
<td>Slave</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>14</td>
<td>Ensure the slave (host1) has caught up</td>
<td>ls</td>
<td>Slave</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>15</td>
<td>Switch master back to host1</td>
<td>switch to host1</td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
</tbody>
</table>

**Note**

With any schema change to a database, the database performance should be monitored to ensure that the change is not affecting the overall dataservice performance.

### 4.11.5. Changing Configuration

Changes to the configuration should be made with `tpm update`. This continues the procedure of using `tpm install` during installation. See Section 7.9.5.17, “tpm update Command” for more information on using `tpm update`.

### 4.11.6. 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 Continuent Tungsten 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/shunning 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>Shun slave host2</td>
<td>datasource host2 shun</td>
<td>Master</td>
<td>Shunned</td>
<td>Slave</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>3</td>
<td>Stop all services on <strong>host2</strong>.</td>
<td>stopall</td>
<td>Master</td>
<td>Stopped</td>
<td>Slave</td>
</tr>
<tr>
<td>4</td>
<td>Update the JVM</td>
<td></td>
<td>Master</td>
<td>Stopped</td>
<td>Slave</td>
</tr>
<tr>
<td>5</td>
<td>Start all services on <strong>host2</strong> slave.</td>
<td>startall</td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>6</td>
<td>Recover slave back</td>
<td>datasource host2 recover</td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>7</td>
<td>Ensure the slave (<strong>host2</strong>) has caught up</td>
<td>ls</td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>8</td>
<td>Shun slave <strong>host3</strong></td>
<td>datasource host3 shun</td>
<td>Master</td>
<td>Slave</td>
<td>Shunned</td>
</tr>
<tr>
<td>9</td>
<td>Stop all services on <strong>host3</strong>.</td>
<td>stopall</td>
<td>Master</td>
<td>Slave</td>
<td>Stopped</td>
</tr>
<tr>
<td>10</td>
<td>Update the JVM</td>
<td></td>
<td>Master</td>
<td>Slave</td>
<td>Stopped</td>
</tr>
<tr>
<td>11</td>
<td>Start all services on <strong>host3</strong> slave.</td>
<td>startall</td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>12</td>
<td>Recover slave back</td>
<td>datasource host3 recover</td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>13</td>
<td>Ensure the slave (<strong>host3</strong>) has caught up</td>
<td>ls</td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
<tr>
<td>14</td>
<td>Switch master to <strong>host2</strong></td>
<td>switch to host2</td>
<td>Slave</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>15</td>
<td>Shun <strong>host1</strong></td>
<td>datasource host1 shun</td>
<td>Shunned</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>16</td>
<td>Stop all services on <strong>host1</strong>.</td>
<td>stopall</td>
<td>Stopped</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>17</td>
<td>Update the JVM</td>
<td></td>
<td>Stopped</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>18</td>
<td>Start all services on <strong>host1</strong> slave.</td>
<td>startall</td>
<td>Slave</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>19</td>
<td>Recover <strong>host1</strong> back</td>
<td>datasource host1 recover</td>
<td>Slave</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>20</td>
<td>Ensure the slave (<strong>host1</strong>) has caught up</td>
<td>ls</td>
<td>Slave</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>21</td>
<td>Switch master back to <strong>host1</strong></td>
<td>switch to host1</td>
<td>Master</td>
<td>Slave</td>
<td>Slave</td>
</tr>
</tbody>
</table>
Chapter 5. Tungsten Connector

Tungsten Connector acts as a proxy service, sitting between client applications and datasources in order to balance the load and provide high availability (HA) support. The service works by accepting raw packets from clients, and then forwarding them to the datasource. The process is reversed when packets are sent back from the datasource and then redirected back to the clients.

Figure 5.1. Tungsten Connector Basic Architecture

In addition to this basic structure, Tungsten Connector also works with the other components of Continuent Tungsten to handle some specific scenarios and situations:

- The connector works in harmony with the Tungsten Manager as part of Continuent Tungsten and enables the connector to redirect queries between known datasources within a given dataservice. For example, when the manager identifies a failed datasource, queries to that datasource are redirected to an alternative datasource without the application being aware of the change.

- The connector works with the Continuent Tungsten configuration and a number of implied or explicit directives that enable the connector to redirect requests within different datasources within the network. For example, the connector can be configured to automatically forward write requests to a database to the active master within the dataservice and reads to active slaves.

Throughout this process the connector is redirecting the network packets sent by application servers to the appropriate host. The contents and individual statements are not processed or accessed. At all times applications and clients using the connector do not need modification as to them it will appear as a MySQL server.

5.1. Connector/Application Basics

Within any typical database deployment there will be two primary considerations:

- High Availability — redirecting requests to alternative servers in the event of a failure.
- Scalability — distributing reads and writes across servers in a replication architecture.

Within a typical basic database deployment clients and application servers will connect directly to databases, as shown in Figure 5.2, "Basic MySQL/Application Connectivity".
The problem with this deployment model is that it is not able to cope with changes or problems. If one or more of your database servers fails, then the application servers must be reconfigured individually to point to an alternative server. In addition, when considering scalability, there is no provision for redirecting reads and writes to masters or slaves.

In an advanced application deployment, individual servers may have been configured to connect to masters and slaves and configured your application to talk directly to the master and/or slaves within the database infrastructure to handle the scalability offered by using replication (Figure 5.3, "Advanced MySQL/Application Connectivity"). For this to work the application must have been modified and be read/write aware, and it must have been configured to manually connect to the different databases according to the operation being performed.
Although this handles the read/write splitting, enabling servers to write to the master and read from one or more slaves, changes to this architecture and structure are not handled. If the master fails, application servers must be manually updated to direct their queries to an alternative host.

When deploying Continuent Tungsten the connector takes over the role of primary connection from your application to the database server, and it handles the redirection of requests to the appropriate database server. Depending on your application configuration and architecture, the connector can be used in two ways:

- As a complete solution for redirecting queries between the master and slave hosts within a dataservice, including HA events.
- As an HA solution redirecting queries to the master and slaves within a dataservice, but with application driven master/slave selection.

When deploying your application with Continuent Tungsten through Tungsten Connector, the application server connectivity is through the connector. The connector takes on the role of primary connection for all requests, while routing and distributing those requests to the active datasources within the dataservice.

The Tungsten Connector is located between the clients and the database servers, providing a single connection point, while routing queries to the database servers. In the event of a failure, the Tungsten Connector can automatically route queries away from the failed server and towards servers that are still operating. During the routing process, Tungsten Connector communicates with the Tungsten Manager to determine which datasources are the most up to date, and their current role so that the packets can be routed properly.
Because the connectivity is between the application service and the Tungsten Connector, the connection to the Tungsten Connector remains constant. Changes to the datasources, including failures, role changes, and expansion or removal of datasources from the dataservice do not require any modification of the application configuration or operation.

Figure 5.5. Tungsten Connector during a failed datasource

For example, in Figure 5.5, "Tungsten Connector during a failed datasource", the slave datasource has failed. While this would break the connection between the Tungsten Connector and the datasource, the connection between the application and Tungsten Connector remains available, and Tungsten Connector will re-route queries to an available datasource without reconfiguring the application server connectivity.

5.2. Routing Methods

Tungsten Connector routes connections between client connections and datasources using a number of different routing methods. These routing methods affect how client applications and datasources are connected to each other, and control the level of inspection by Tungsten Connector of the connections and statements as they pass through the connector service.

The Tungsten Connector works with Tungsten Manager to automatically route clients connected to the connector to an appropriate server, balancing the load when communicating with slaves. The different methods are involved in effective read/write splitting, i.e. the ability to correctly route requests to the masters or slaves within the network according to the type of operation being performed by the client. This can be performed automatically, or manually, or through a series of specific configurable routing methods.

Figure 5.6. Tungsten Connector routing architecture
Routing selection is made by the connector based on the availability information using a combination of different settings and parameters. Each level overrides or augments the previous level, and each can be specified in different locations, such as the `user.map`, connecting string, or within individual supplied statements. The settings are processed in the order shown below; later setting override earlier settings.

For example, selecting the SQL routing method defines the default behavior. Specifying the QoS in the `user.map` file supercedes the SQL routing; setting QoS in a comment before the SQL statement supercedes the user and default behavior. Specifying an affinity in the comments overrides both the user and default configuration settings.

- **Selected routing method**, see Section 5.2.1, "Connector Routing Methods"
- **Quality of Service (QoS) specification**, see Section 5.2.2, "Connector Quality of Service (QoS) Selection"
- **Load balancer selection** (implied by QoS), see Section 5.2.3, "Connector Load Balancers"
- **Slave latency**, including optional maximum latency setting, see Section 5.2.4, "Specifying Required Latency"
- **Affinity specification**, see Section 5.2.5, "Setting Host Affinity and Direct Reads"

These different routing configurations can be selected according to the global configuration, and customization at different points in the communication channel. For example, SQL-based routing configures basic load-balancing, but allows SQL comments to be used to change the default QoS mode and affinity.

<table>
<thead>
<tr>
<th>Routing Method</th>
<th>QoS</th>
<th>Latency</th>
<th>Affinity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Configuration</td>
<td>Yes</td>
<td>Implied</td>
<td>Yes</td>
</tr>
<tr>
<td>Connection String</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><code>user.map</code></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SQL statement</td>
<td>Yes (with SQL routing enabled)</td>
<td>Yes (with SQL routing enabled)</td>
<td>No</td>
</tr>
</tbody>
</table>

At all times, the connector uses the current status of the MySQL servers to make decisions about where queries and connections should be routed. Changes to the master, and availability or accessibility of individual dataservers will always be taken into account when routing the queries. For information on what happens if failure occurs during an operation or transaction, see Section 5.5, "Connector Operational States".

The routing methods can either involve direct reads, SmartScale, host-based routing, or SQL inspection-based routing to redirect reads and writes to the appropriate server. In addition to these implied routing methods, clients can also specifically select which host to communicate with through the use of tags and options provided through the connection string.

The selection of a datasource occurs at the point the client connects, and this datasource connection choice remains in effect until the client disconnects, unless a failover or switch occurs.

### 5.2.1. Connector Routing Methods

The supported routing methods, typical uses and use cases are listed in Table 5.1, "Routing Method Selection".

<table>
<thead>
<tr>
<th>Routing Method</th>
<th>Host Selection</th>
<th>Auto R/W Splitting</th>
<th>Slave Latency</th>
<th>maxAppliedLatency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smartscale</td>
<td>By Session</td>
<td>Yes (by SQL statement)</td>
<td>Lazy</td>
<td>Yes</td>
</tr>
<tr>
<td>Direct Reads</td>
<td>By Content</td>
<td>Yes (by SQL statement)</td>
<td>Lazy</td>
<td>Yes</td>
</tr>
<tr>
<td>Host-based</td>
<td>By Hostname</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Port-based</td>
<td>By Network Port</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>SQL-based</td>
<td>By SQL comment</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Important**

Both SmartScale and R/W splitting cannot be enabled at the same time. This is because they are two sides of the basic functionality. R/W splitting and SmartScale both use SQL introspection to determine whether a query should be directed to a master or a slave. SmartScale combines this with an intelligent load-balancer. R/W splitting uses a simpler direct redirection.

In addition to the selection and configuration mechanisms supported, a routing method should be chosen based on your application abilities.
• If the application is replication-aware, and can already direct queries to master or slaves based on the operation type, use Section 5.2.9, "Host-based Routing" or Section 5.2.10, "Port-based Routing".

• If the application has full control of the SQL statements submitted (i.e. not through a third-party tool, or Object-Relational Modeling library), and can already direct queries to master or slaves based on the operation type, use Section 5.2.8, "SQL Routing".

• If the application uses non-auto-commit statements (for example, Hibernate), Section 5.2.9, "Host-based Routing", or Section 5.2.8, "SQL Routing".

• If the application does not fit any of these categories, or is not replication aware use either either Section 5.2.7, "Direct Routing" or Section 5.2.6, "Smartscale Routing".

5.2.2. Connector Quality of Service (QoS) Selection

Depending on the chosen routing and authentication method, the 'Quality of Service' (QoS) setting can be specified as part of the SQL statement, host, or user configuration, and affects the selection of the MySQL server:

• **RO_RELAXED** [132]
  This setting enables the connector to redirect the query as if it were read-only, and therefore prefer a slave over a master, but will choose a master if no slave is available.

• **RW_STRICT** [132]
  This setting indicates that the query is a write and should be directed to a master.

• **RW_SESSION**[132]
  Where possible, the QoS should be set for read/write splitting according to the current session state.

These hints for the connection can be set, for example by using the value in the comments during SQL routing, or by setting the corresponding QoS value in the user.map file.

Further, connectivity can be influenced by setting a suitable latency value, or an explicit affinity. This information can be specified either within the connection strings, within the user.map, or through configuration.

The rules for selection of whether a connection is made to a master or a slave is therefore controlled by comparing all of these settings and the selecteds routing mechanism together.

<table>
<thead>
<tr>
<th>SmartScale</th>
<th>QoS</th>
<th>maxAppliedLatency</th>
<th>Selection Order</th>
<th>Affinity</th>
<th>Master Selected</th>
<th>Slave Selected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabled</td>
<td>Not Specified</td>
<td>Not Specified</td>
<td>Not Specified</td>
<td>Slave:Master</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Not Enabled</td>
<td><strong>RO_RELAXED</strong> [132]</td>
<td>Not Specified</td>
<td>Not Specified</td>
<td>Slave:Master</td>
<td>Only if no slave available</td>
<td>Yes</td>
</tr>
<tr>
<td>Not Enabled</td>
<td><strong>RO_RELAXED</strong> [132]</td>
<td>Specified</td>
<td>Not Specified</td>
<td>Slave:Master</td>
<td>Only if no slave available</td>
<td>Only if slave latency &lt; maxAppliedLatency</td>
</tr>
</tbody>
</table>

5.2.3. Connector Load Balancers

The load balancing model used, according to the selected QoS is defined by a number of different load balancing classes. These are configured automatically when different QoS is selected, be explicitly changed by altering the configuration file. The supported load balancers are detailed in the table below:

<table>
<thead>
<tr>
<th>Load Balancer</th>
<th>Default QoS</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefaultLoadBalancer</td>
<td><strong>RW_STRICT</strong> [132]</td>
<td>Always selects the master data source</td>
</tr>
<tr>
<td>MostAdvancedSlaveLoadBalancer</td>
<td><strong>RO_RELAXED</strong> [132]</td>
<td>Selects the slave that has replicated the more events, by comparing data sources &quot;high water&quot; marks. If no slave is available, the master will be returned.</td>
</tr>
<tr>
<td>LowestLatencySlaveLoadBalancer</td>
<td></td>
<td>Selects the slave data source that has the lowest replication lag, or appliedLatency in ls -l within cctl output. If no slave data source is eligible, the master data source will be selected.</td>
</tr>
<tr>
<td>RoundRobinSlaveLoadBalancer</td>
<td></td>
<td>Selects a slave in a round robin manner, by iterating through them using internal index. Returns the master if no slave is found online</td>
</tr>
<tr>
<td>HighWaterSlaveLoadBalancer</td>
<td><strong>RW_SESSION</strong> [132]</td>
<td>Given a session high water (usually the high water mark of the update event), selects the first slave that has higher or equal high water, or the master if no slave is online or has replicated the given session event. This is the default used when SmartScale is enabled.</td>
</tr>
</tbody>
</table>
5.2.4. Specifying Required Latency

Depending on the selecting routing method, load balancer and QoS setting, a slave will automatically be chosen when the host connects. The maximum allowed affinity can be set to limit the connection to only use a slave that is within the specified maximum applied latency limit.

This can be specified in the connection string, and enables slave selection based on the slave which has a latency within the specified limit. For example, using the connection string:

```
jdbc://connector1:3306/database?maxAppliedLatency=5
```

Will specify that a host with a latency of less than 5 seconds should be selected.

The option can be set globally by configuring the JDBC options used by the connector using the `--property=jdbc.driver.options=maxAppliedLatency=5 [244]` option to tpm, or by setting the `jdbc.driver.options` configuration parameter.

5.2.5. Setting Host Affinity and Direct Reads

Affinity enables you to specify at connection time that the connector should forward the connection to a particular host or service for reads, if the service is available. For example, within `user.map`:

```
user password east_west east
```

Defines a user that uses the `east_west` service, but prefers being routed to the `east` Service for reading from a slave.

Affinity can also specified within the connection string:

```
jdbc://connector1:3306/database?affinity=host3
```

The option can be set globally by configuring the JDBC options used by the connector using the `--property=jdbc.driver.options=affinity=host [244]` option to tpm, or by setting the `jdbc.driver.options` configuration parameter.

Affinity can also be combined with other node selection, such as QoS. For example, by combining the affinity and `RO_RELAXED [132]`, then the specified slave will be used first, if the load-balancer setting matches, then another slave within the same service, and finally the master. For example, in a dataservice with three nodes, where `node1` is the master:

```
shell> mysql -h127.0.0.1 -P9999 databasename@qos=RO_RELAXED&affinity=node2
```

Would use `node2` first, then `node3`, and finally `node1` if the others are not available.

**Note**

Within a composite datasource, you cannot specify a specific host. You can only specify a physical dataservice within the composite datasource. For example in a composite service with east and west physical dataservices:

```
shell> mysql -h127.0.0.1 -P9999 databasename@qos=RO_RELAXED&affinity=east
```

Additionally, the `user.map` can be configured to direct specific users to a slave by using the `@direct` keyword. For example, the following line in `user.map` will always direct the user to a slave, ignoring latency and load balancing settings:

```
@direct readme
```

5.2.6. Smartscale Routing

5.2.6.1. Enabling SmartScale Routing

To enable SmartScale routing, configure the dataservice using the `--connector-smartscale` option. The session ID identification should also be specified by using the `--connector-smartscale-sessionid` option:

```
shell> tpm configure alpha \
    --connector-smartscale=true \
    --connector-smartscale-sessionid=DATABASE
```

In this mode, any client application can open a connection to the connector, and queries will automatically be redirected according to the SQL statement content.

In addition, all users that connect to the database must be granted the `REPLICATION CLIENT` privilege so that the user can compare the current replicator progress for session information. This can be granted using:

```
mysql> GRANT REPLICATION CLIENT ON *.* to app_user@'%
```
5.2.6.2. Disabling SmartScale Routing

To disable SmartScale routing if it has been previously configured:

```
shell> tpm configure alpha \\n    --connector-smartscale=false
```

5.2.7. Direct Routing

<table>
<thead>
<tr>
<th>Auto Read/Write Splitting</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master Selection</td>
<td>Automatically, by SQL examination</td>
</tr>
<tr>
<td>Slave Selection</td>
<td>Automatically, by SQL examination</td>
</tr>
<tr>
<td>QoS Compatibility</td>
<td>None</td>
</tr>
<tr>
<td>SmartScale Compatibility</td>
<td>None</td>
</tr>
</tbody>
</table>

Direct routing is a simplified form of SmartScale that uses a highly-efficient automated read/write splitting system, where all auto-committed read-only transactions are routed to a pool of read-only slave datasources. Unlike SmartScale, Direct routing pays no attention to the session state, or replicated data consistency.

This means that performing a write and immediately trying to read the information through a Direct routing connection may fail, because the Connector does not ensure that the written transaction exists on the selected slave.

Direct routing is therefore ideal in applications where:

- Applications perform few writes, but a high number of reads.
- High proportion of reads on 'old' data. For example, blogs, stores, or machine logging information

Where applications are performing writes, followed by immediate reads of this data, for example conferencing and discussion systems, where reading stale data that has recently been written would create significant application failures, the solution should use SmartScale.

Read/Write splitting is supported through examination of the submitted SQL statement:

- If the statement contains `SELECT` and does not contain `FOR UPDATE`, the query is routed to an available slave.
- If the statement starts `SHOW` ... then it is routed to a slave.
- All other queries are routed to the master.

5.2.7.1. Enabling Direct Routing

To enable direct routing, there are two components, the configuration of the cluster, and the `user.map` contents. To enable direct mode the cluster must have been configured with read/write splitting enabled:

```
shell> tpm configure alpha \\
    --connector-rwsplitting=true
```

Users within the `user.map` should be defined using the `@direct` directive. For example:

```
@direct sales
```

In this mode, any client application can open a connection to the connector, and queries will automatically be redirected according to the SQL statement content.

5.2.7.2. Limitations of Direct Routing

- Prepared statements must be enclosed within an explicit transaction boundary in order to be correctly routed to a master. For example:

  ```
  BEGIN
  PREPARE...
  EXECUTE...
  COMMIT
  ```

- Ephemeral objects, including temporary tables, session variables and other objects that are session specific will not be accessible during direct routing.
- Stored procedures that update data in the database should never be called using a basic `SELECT` statement:
Instead, add the `FOR UPDATE` keywords to ensure it is routed to the master:

```sql
mysql> SELECT update_function('data') FOR UPDATE;
```

5.2.8. SQL Routing

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto Read/Write Splitting</td>
<td>No</td>
</tr>
<tr>
<td>Master Selection</td>
<td>Manually, by SQL comments</td>
</tr>
<tr>
<td>Slave Selection</td>
<td>Manually, by SQL comments</td>
</tr>
<tr>
<td>QoS Compatibility</td>
<td>Supported</td>
</tr>
<tr>
<td>SmartScale Compatibility</td>
<td>Yes</td>
</tr>
<tr>
<td>Direct Compatibility</td>
<td>Yes</td>
</tr>
</tbody>
</table>

With SQL-based routing, the redirection of queries and operations through the Connector is controlled by hints on the QoS provided in the comments of individual statements. Note that this is explicit routing using SQL comments, not the automated read/write splitting supported by Direct or SmartScale routing.

Unless otherwise specified, statements will go to the current master to be executed. To specify that a statement can be executed on the slave, place a comment before the statement:

```sql
/* TUNGSTEN USE qos=RO_RELAXED */ SELECT * FROM TABLENAME
```

This indicates to the connector that queries can go to a slave, although they may still be executed on the master.

5.2.8.1. Enabling SQL Routing

To enable SQL routing, use the following operations with `tpm`:

```shell
tpm configure alpha
   --property=selective.rwsplitting=true
```

5.2.8.2. Limitations of SQL Routing

- Read/write splitting must be handled entirely by the client application using the comments to specify which statements are slave safe. Unless applications explicit make the decision to write and read to the hosts using the comment system, operations may go to the wrong hosts.
- Prepared statements must be executed against the master.
- When testing the operation of the read/write splitting through the `mysql` client, ensure that command-line client is called using the `-c` option to ensure that comments are preserved:

```shell
mysql -c -h host -u tungsten -ppassword -P9999 test
```

5.2.9. Host-based Routing

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto Read/Write Splitting</td>
<td>No</td>
</tr>
<tr>
<td>Master Selection</td>
<td>Manually, by hostname/IP address</td>
</tr>
<tr>
<td>Slave Selection</td>
<td>Manually, by hostname/IP address</td>
</tr>
<tr>
<td>QoS Compatibility</td>
<td>None</td>
</tr>
<tr>
<td>SmartScale Compatibility</td>
<td>None</td>
</tr>
</tbody>
</table>

Host-based routing uses specific hostnames to provide the distinction between read and write availability within the connector. Two different hostnames and associated IP addresses need to be created on each connector host. Clients connecting to one host will be routed to the current master for writing, and connections to the other host will be redirected to a current slave using the current load-balancing algorithm.

Once enabled, a client can open a connection directly to a master or slave by connecting to the appropriate IP address or hostname. For example:

```shell
mysql -h master.localhost
```

Will connect to the currently active master, while:
Would connect to any currently available slave.

5.2.9.1. Enabling Host-based Routing

To enable host-based routing requires both operating system and Connector based configuration changes:

1. The following steps must be made to the operating system configuration for each Connector host that will be configured within the dataservice:
   a. Add a second IP address to the host. This can be achieved either by adding or exposing a second physical ethernet device, or by exposing an alias on an existing hardware interface.
      
      For example, to add a second IP address to the physical `eth0` interface:
      
      ```shell
      sudo ifconfig eth0:1 192.168.2.24
      ```
      
      To ensure this is retained during a restart, update your network configuration with the additional physical interface and IP address.
   b. Update the `/etc/hosts` file to reflect both addresses and appropriate hostnames. For example:
      
      ```
      192.168.2.20 host1 master.host1
      192.168.2.21 slave.host1
      ```
   c. When using DNS to resolve addresses, the DNS should also be updated with hostnames to match those configured for each IP interface.

2. Update or configure your dataservice by listing the master and slave hostnames for each host using the `--connector-rw-addresses` and `--connector-ro-addresses` respectively. For example:

   ```shell
   shell> tpm configure alpha 
   ... \ 
   --connector-rw-addresses=master.host1,master.host2,master.host3 \ 
   --connector-ro-addresses=slave.host1,slave.host2,slave.host3
   ```

   Once configured, client applications must be configured to select the appropriate host based on the operation they are performing.

5.2.9.2. Limitations of Host-based Routing

- Prepared statements must be executed on the master.
- Smartscale cannot be enabled at the same time as host-based routing.
- QoS selection will not be honored.

5.2.10. Port-based Routing

<table>
<thead>
<tr>
<th>Feature</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto Read/Write Splitting</td>
<td>No</td>
</tr>
<tr>
<td>Master Selection</td>
<td>Manually, by network port</td>
</tr>
<tr>
<td>Slave Selection</td>
<td>Manually, by network port</td>
</tr>
<tr>
<td>QoS Compatibility</td>
<td>None</td>
</tr>
<tr>
<td>SmartScale Compatibility</td>
<td>None</td>
</tr>
</tbody>
</table>

Port-based routing configures two independent ports that enable client applications to select whether to connect to a master or slave based on the port they connect to. This method relies on the application choosing the correct port, automatic r/w splitting is not supported. Similar to host-based routing, port-based routing requires the client application to be modified to manually select the appropriate port.

Once enabled, a client can open a connection directly to a master or slave by connecting to the appropriate port. For example:

```shell
shell> mysql -P13306
```

Will connect to the currently active master, while:

```shell
shell> mysql -P13307
```

The ports to be used for each connection type are configurable during installation.
5.2.10.1. Enabling Port-based Routing

Enabling port-based routing requires configuring the two ports that will accept queries. One port will be designated as the master port, one the read-only port, and queries will be automatically routed accordingly. For example:

```
shell> tpm configure alpha \
... \n--connector-listen-port=3306
```

Client applications must be updated to support the two port interfaces and manually direct their queries to the appropriate master or slave.

Using port routing in this way effectively marks all connections to the read-only port as behaving in a similar fashion to setting the connection QoS to `RO_RELAXED` [132].

5.2.11. Read-only Routing

It is possible to deploy a connector that has been configured to provide read-only access to the underlying databases on the standard port. This enforces read-only connectivity through this connector, regardless of any session or connector configuration options. This can be useful if a standalone connector, or single connector within a dataservice.

To enable this functionality, configure the connector using the `--connector-readonly=true` [225]:

```
shell> tpm configure alpha \
... \n--connector-readonly=true
```

5.3. Using Bridge Mode

Bridge mode eliminates the need to create or define users and passwords within the `user.map` file. Instead, the connector acts as a router connecting the network sockets between client application and MySQL servers.

Figure 5.7. Tungsten Connector Bridge Mode Architecture

Bridge mode provides a simpler method for connecting clients to MySQL, but with reduced facilities, as outline in the table below:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Standard Mode</th>
<th>Bridge Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master/Slave Selection</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Switch/Failover</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Read/Write Splitting</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Seamless Reconnects</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

Bridge mode connections operate as follows:

1. Client opens network connection to Connector
2. Connector allocates a network buffer for the client network connection to the database server.
3. Connector opens a network connection to a database server based on the connection parameters (master/slave selection).
4. Connector allocates a network buffer for the database server to the client application.
5. Connector directly attaches the network sockets sockets together.

Because the network sockets between the two sides are connected directly together, the following behavior applies to bridge mode connections:

- User authentication is handled directly by the database server, rather than through the `user.map` file.
- In the event of a failover or switch of the database servers, all active connections to the affected servers are closed.
- Smartscale and packet inspection to provide read/write splitting are not supported, since the Connector does not access individual packet data.

### 5.3.1. Enabling Bridge Mode

To enable Bridge Mode, the `--connector-bridge-mode` option to `tpm` must be set to `true`:

```
shell> tpm configure alpha --connector-bridge-mode=true
```

The default value is `false`, i.e. use the connector in non-bridge mode with full support for read/write splitting and operation.

Bridge Mode can also be used with read-only database servers and affinity if required.

In addition to enabling and disabling Bridge Mode, the size of the buffer can also be set by using the `bridgeServerToClientBufferSize` and `bridgeClientToServerBufferSize` parameters. This configures the size of the buffer used to hold packet data before the packet is forwarded.

The default size is 1024 bytes (1KB).

The default size is 262144 bytes (256KB).

A buffer is opened in each direction for each connection made to the connector when operating in bridge mode.

The total memory allocated can be calculated using the formula:

```
(connections * (bridgeClientToServerBufferSize + bridgeServerToClientBufferSize))
```

For example, with the default settings, 20 simultaneous connections will require 40KB of RAM to service the buffers.

For example, with the default settings, 20 simultaneous connections will require 10MB of RAM to service the buffers. With the default connector heap size the Connector should be able to handle up to 500 simultaneous connections.

### 5.4. User Authentication

When configuring Tungsten Connector it is important to ensure that you have a `user.map` in place. The role of `user.map` is to define the usernames and passwords of users that will be connecting to the dataserver.

There is no authentication within the connector. Instead, the connector sends authentication information onto the dataserver. However, the MySQL network protocol exchanges a token between the client and the dataserver in order to authenticate the connection and is designed to prevent 'man in the middle' attacks.

Unfortunately, 'man in the middle’ is exactly how Tungsten Connector operates, as the man in the middle to redirect queries to different dataservers as the list of active dataservers changes during the operation of a cluster. The authentication exchange cannot be reinitiated by the dataserver and client, so the Tungsten Connector performs this authentication exchange on behalf of the client using the user and password information from a special file called `user.map`.

Figure 5.8. Tungsten Connector Authentication
To get round this limitation, the connector operates as follows:

- Client opens a connection to the connector and authenticates.
- Connector connects to the datasource using the username supplied by the client, and the corresponding password stored within `user.map`.
- Database server returns the authentication token to the connector.
- Connectors sends the same authentication token back to the client.

This process gives the client application the authentication token required to enable it to communicate with the dataserver and the same token to be used by the connector.

For this system to work, a file, `user.map`, must exist on every connector installation, and it must contain the information for all users that will connect to the datasources from each client. Without this information, connectors will be unable to login on behalf of the client applications.

**Important**

All the users that require access to your MySQL servers through the Tungsten Connector must have an entry in the `user.map`. Without this information, the Tungsten Connector has no way of providing an onward connection to a MySQL server.

The `user.map` file primary role is to operate as the source for authentication information within the connector. However, through the use of additional flags and keywords, the file can also define the routing methods used by different users when connecting to datasources, and different dataservices.

### 5.4.1. `user.map` File Format

The current `user.map` file is located within the `tungsten-connector/conf` directory within an active installation. The file should be synchronized across all the servers within a dataservice. For more information on methods for keeping the file in sync, see Section 5.4.6, “Synchronizing `user.map` Data”.

The `user.map` file contains the usernames and passwords for each user that connects to the connector and the downstream MySQL server, and these entries are required for authentication. If an entry does not exist within `user.map` users will be unable to connect to MySQL through the connector.

**Important**

All the users that require access to your MySQL servers through the Tungsten Connector must have an entry in the `user.map`. Without this information, the Tungsten Connector has no way of providing an onward connection to a MySQL server.

The rules for the format of the file are as follows:

- Anything after a `#` (hash) symbol are interpreted as comments and ignored. For example:

  ```
  # This line is a comment
  ```

- The following character cannot be used as the username, password or dataservice values:

  ```
  space | # pipe \
  | # tab
  ```

- If direct reads (using the `@direct` directive, the following characters should be avoided within passwords:

  ```
  & # ampersand
  @ # at sign
  = # equals
  ? # question mark
  ```

- Using the – (hyphen) character as a password indicates that there is an empty or no password (""") for the specified user.

The basic format for user entries within the `user.map` is:

```
username password servicename [affinity]
```

Where:

- `username` — the username to be used for authentication.
The `username` also provides hooks into additional options; see `@script`, `@direct`, `@hostoption`.

- **password** — the password to be used for authentication.
- **servicename** — the name of the dataservice or composite service to which this username/password apply.
- **affinity** — if the servicename is a composite service, the `affinity` identifies which service should be preferred for reads.

For example, to configure the user `sales` with the password `secret` to use MySQL servers within the `alpha` dataservice:

```plaintext
sales secret alpha
```

To configure a user that has no password:

```plaintext
sales - alpha
```

To configure a user within a composite service:

```plaintext
sales secret nyc_sfo
```

To configure a user within a composite service, preferring the `sfo` service for read-only connections:

```plaintext
sales secret nyc_sfo sfo
```

### 5.4.2. `user.map` Direct Routing

To enable direct reads, as defined within Section 5.2.7, “Direct Routing”, the entries for the user within `user.map` must be prefixed using the `@direct`. For example:

```plaintext
@direct sales
```

Note that the standard user, password and service must be defined:

```plaintext
sales secret alpha
@direct sales
```

For limitations of direct routing, see Section 5.2.7.2, “Limitations of Direct Routing”.

### 5.4.3. `user.map` Host Options

The `user.map` provides a configuration point to enable the connector to support host options that enable you to define qualities of service against specific hosts, as configured according to the guidance within Section 5.2.9, “Host-based Routing”.

When configuring the host options, the hostnames must have previously been defined and be resolveable.

The QoS within `user.map` has the following format:

```plaintext
@hostoption hostname QoS
```

For example, to enable `RW_STRICT` on one host and `RO_RELAXED` on the other:

```plaintext
@hostoption readwrite.master qos=RW_STRICT
@hostoption readonly.master qos=RO_RELAXED
```

### 5.4.4. `user.map` Updates

When the `user.map` file is updated:

- The Tungsten Connector should automatically identify that the file has been changed and reload the file, updating the user map.
- To manually force the users to be updated, for example, if the `user.map` uses the `@script`, use the `tungsten flush privileges` command:

```sql
mysql> tungsten flush privileges;
+--------------------------------+
| Message                        |
+--------------------------------+
| user.map reloaded successfully |
+--------------------------------+
1 row in set (0.00 sec)
```

- When using `@direct` entries in `user.map`, the connector may need to be restarted using `connector restart`: 
This will disconnect all connected clients, but the connector itself should be unavailable only for a short time.

- When the connector installation is updated using `tpm`, for example during an upgrade, the `user.map` and `dataservices.properties` are automatically copied into the new installation automatically and do not need to be manually copied or update.

To perform an update without automatically copying the `user.map` file using the `--connector-delete-user-map` option to `tpm`.

### 5.4.5. Generating `user.map` Entries from a Script

The content of the `user.map` file can be generated automatically, for example by automatically extracting information from a separate service, such as LDAP, NIS or others. To specify the script that will generate the information, the `@script` directive can be used within the `user.map`:

```bash
@script /opt/continuent/share/usermap
```

When using the script method:

- The information must be generated in the same format as for standard entries, i.e.:
  ```text
  username password servicename
  ```

- If the script generates multiple entries with the same name, the later output will overwrite the previous entry.

- Multiple `@script` directives can be specified. Each will be processed in turn.

- If a generated list of usernames changes due to the scripts, the connector must be manually forced to reload the `usermap` using `tungsten flush privileges` on a connector connection.

- If the file is placed into `/opt/continuent/share` then the script will be retained during upgrades through `tpm update`.

- If a script within the `@script` fails to be executed correctly, or generates no user entries, the connector will fail to start.

The script itself can be relatively simple, the standard output of the command must contain the user entries to be included in `user.map`. Standard error is ignored.

For example:

```bash
#!/bin/bash
echo 'app_user password dsone'
```

This generates a simple user entry.

### 5.4.6. Synchronizing `user.map` Data

Continuent Tungsten does not automatically synchronize information contained within the `user.map` across all the nodes within the cluster. The connector does not identify, track, or update `user.map` content when it sees password changes.

Instead, the file must be updated by hand, through the `@script` directive, and synchronized across multiple hosts either manually or by using a script. For example:

```bash
#!/bin/bash
for HOST in host1 host2 host3 host4
do
  rsync /opt/continuent/tungsten-connector/conf/user.map \ 
  $HOST:/opt/continuent/tungsten-connector/conf/user.map
done
```

If `@script` directives, the corresponding scripts must also be included within this synchronization step.

---

**Important**

All servers within the cluster must have an identical `user.map` configuration. Failure to have a synchronized configuration may lead to clients being unable to connect to the connector and database servers.

### 5.4.7. `user.map` Limitations

The `user.map` configuration has the following limitations:
• Users must be defined for each dataservice; if there is a common user that can be used in any of your configured dataservices, there must be an individual line for each dataservice. For example:

  sales secret alpha
  sales secret beta
  sales secret gamma

• When using user.map with multiple dataservices, additional data services must exist in dataservices.properties. Only add the physical data services you would like to work with. Any composite data services will automatically be discovered. The connector must be restarted once the data services have been added.

• Specifying a composite dataservice that has not been defined will raise an error.

• If the user.map contains multiple entries for the same user, only the last entry will be used.

5.4.8. Host-based Authentication

In addition to the explicit user/host based authentication support, the connector also includes general host-based authentication that allows client connections only from specific hosts.

To enable host-based authentication, create a file authorized_hosts within the cluster-home/conf directory of the active installation. The location of the file can be explicitly set by updating the value of the authorized.hosts.file parameter within the connector.properties:

```
authorized.hosts.file=../../tungsten-connector/conf/authorized_hosts
```

**Important**

The authorized_hosts file is not automatically distributed during deployment and updates. The file must be manually copied to other hosts.

If the file exists, host-based authentication is enabled. With an empty, all client connections are denied. The format of the file is that each line defines the host address and netmask in CIDR format. For example:

```
192.168.1.0/24
```

Enables connectivity from all hosts in the range 192.168.1.0-192.168.1.255.

Host-based authentication is disabled if the file does not exist or is unreadable, and all hosts will be able to access the connector (as in the default installation). If the file exists,

5.5. Connector Operational States

During operation, the connector goes through a number of different states and state transition during specific events. The default mode is the **Online** state, where the connector operates as configured.

During operation, all configured connectors within the dataservice remain in contact with the manager, see Section 5.6, “Connector/Manager Interface” for more information.

Supported states by the Connector are:

• **Online** State
  
The Connector operates as configured, redirecting connections to the corresponding master or slave.

• **On Hold** State
  
  In this state, the connector will continue as normal allowing existing connections to continue until the delayBeforeOfflineIfNoManager is reached. New connections are paused.

• **Offline** State
  
  When the connector enters the offline state, the connector terminates all connections, and blocks all new connections.

The state of a connector can be modified by using the router command within cctrl. This can be used to manually place the connector into online or offline states. For example, to put a connector online the full host and process ID must be used:

```
cctrl> router connector@host1[22476] online
```

Wildcards can be used to enable or disable all the hosts. For example, to place all connectors online:
While in Automatic policy mode, connectors will automatically be placed online if they have entered the Offline state as part of a failover. If the routers have been manually placed offline, routers must be manually placed back online.

While in the Online state, the connector behaves and alters its operation according to the following states and events:

### 5.5.1. Connections During Automatic Failure/Failover

When an automatic failure or failover is identified, for example when the dataservice is in the Automatic policy mode, and the master is automatically switched to a new host, the following sequence occurs:

1. All connections to the failed datasource are terminated immediately. This ensures that running transactions or operations are terminated by the database server.
2. Connections to clients will remain open and be reconnected transparently, providing they are not within a transaction. For more information, see Section 5.5.3, "Connections During Connection Failures".

   Only if there is a problem with the connection or an I/O error will the problem be forwarded to the clients.

As with a direct database connection, the client application should handle the reconnection to the Connector, which will be then be redirected to the corresponding master or slave datasource.

### 5.5.2. Connections During Manual Switch/Failover

When a manual failure has been initiated, for example, during a datasource shun, or switch operation has been initiated, the Connector follows this sequence:

1. New connection attempts to the failed datasource are suspended; this gives the impression of a 'hung' connection that must be managed by the client application through the normal timeout procedure.
2. Existing connections to the datasource are terminated under two conditions:
   - As the connections are naturally closed.
   - Open connections are forcibly disconnected after the timeout specified by the waitForDisconnectTimeout parameter. By default, this is 5 seconds. To eliminate waiting, the waitForDisconnect parameter can be set to false.

   Once either condition has been met, any remaining connections are closed.
3. New connections (including re-connections) are enabled, and will be routed to the appropriate master or slave.

Client applications should be configured to reconnect to the connector with an interval larger than the disconnect timeout within the connector. This will ensure that the client reconnects when the connector is able to accept the new connection.

### 5.5.3. Connections During Connection Failures

In the event of a connection failure between a running datasource and the connector, and providing the connection is deemed idle, the connector will transparently reconnect to the failed datasource when the following conditions have been met:

- The connection is not executing any requests.
- The connection is not in the middle of a transaction.
- No temporary tables have been created during this connection.

If all three conditions are met, a new connection will be opened. Connections between the client and the connector will be unaffected.

This option is enabled by default. To disable transparent reconnections, use --connector-autoreconnect=false option to tpm during installation.

### 5.5.4. Other Errors

The Connector attempts to emulate and effectively represent any errors raised by the datasource to which the connector has routed the client connection.

- The Tungsten Connector uses the Tanuki Java Service Wrapper to manage the running process. If the Connector process fails, the service wrapper will automatically restart it. If the connector fails repeatedly, attempts to restart will be stopped. The status and reason for these failures can be tracked by examining the connector.log log file.
Connected client applications will be terminated, but should be able to reconnect once the Connector has been restarted.

- Database errors, including invalid statements, operations, or security failures, will be represented identically by the Connector to any clients.

### 5.5.5. Connector Keepalive

Connections to MySQL servers can automatically time-out according to the `WAIT_TIMEOUT` variable configured within the MySQL server.

To prevent these connections being automatically closed, the connector can be configured to keep the connection alive by submitting a simple `SELECT` statement (actually `SELECT 'KEEP_ALIVE'`) periodically to ensure that the MySQL timeout is not reached and the connection closed.

Two parameters configure the keepalive functionality:

- `connection.keepAlive.interval`
  
  The interval used to check for idle connections. If set to a value of 0, the keep alive check is disabled. Any value greater than zero is the interval check period in seconds.

- `connection.keepAlive.timeout`
  
  The keep-alive statement is submitted if the time since the last activity reaches this timeout value.

The default setting for both parameters is 0 (disabled).

```shell
shell> tpm update alpha \
    --property=connection.keepAlive.interval=autodetect \
    --property=connection.keepAlive.timeout=autodetect
```

When set to `autodetect`, the values are automatically calculated by the connector based on the `WAIT_TIMEOUT` value configured in the MySQL server.

```java
connection.keepAlive.interval = (int) Math.floor(wait_timeout * 0.10);
connection.keepAlive.timeout = (int) Math.floor(wait_timeout * 0.7);
```

These calculations cannot be modified, but the properties can be explicitly set by using the `--property` to explicitly set the property through `tpm`, for example:

```shell
shell> tpm update alpha \
    --property=connection.keepAlive.interval=30 \
    --property=connection.keepAlive.timeout=15
```

### 5.6. Connector/Manager Interface

The connector remains in constant communication with the Tungsten Manager during operation. This enables the connector to respond to failures and errors, whether automatically identified, or manually triggered. For example, when a manual switch operation occurs, the manager communicates this information to all of the connectors. Each connector then responds according to the rules outlined in Section 5.5, “Connector Operational States”.

The connector remains in communication with one, and only one, manager at a time. If the manager becomes unavailable, connector tries to communicate another manager within the dataservice.

Communication from the manager to the connectors is made in parallel using multiple threads, this ensures that all connectors are made aware of a change in the topology of the cluster at the same time, rather than a round-robin or staged distribution. When a change has been requested, the manager waits for a response from the cluster before confirming that switch and operational change has taken place.

Communication between managers and the connectors is handled on ports 11999 (managed by `--router-gateway-port`) and 12000 (managed by `--mgr-rmi-remote-port`). The connection is used to exchange cluster status and individual datasource availability as identified by the manager so that decisions about active connections can be effectively made by the connector.

In the event that the connection between the connector and the manager is broken, the connector enters a fail-safe mode called `On Hold`. In this state, connections to and from the connector and datasources will continue as normal until a timeout, configured by the `delayBeforeOfflineIfNoManager` property, is reached. By default, this timeout is 600 seconds (10 minutes). Once the timeout has been reached, the connector reaches the `Offline` state.

All of the information about the dataservice, including the other nodes, topology and individual node states and roles are entirely determined by the Connector by requesting this information from the Manager. No on-disk record or description is stored or stored,
created, or read by the Connector. When the Connector is first started, it connects to a manager and requests the full cluster configuration.

If the Connector cannot communicate with a manager, the connector remains in the offline state until a manager can be reached.

5.7. Clients and Deployment

5.7.1. Using with the Command Line

5.7.2. Using with PHP

5.7.3. Using with Java

5.7.4. Using with Ruby

5.7.5. Client Interface

5.7.6. Using with EC2 Elastic Load Balancer

5.7.7. Using with HA Proxy

Tungsten Connector can be used in combination with an HA Proxy installation to provide a high-availability connection to the underlying connectors that then provide an intelligent connection to the datasources within the cluster.

A suitable configuration can be added to a basic HA Proxy installation using the following settings:

```
# backend
#---------------------------------------------------------------------
listen connector
    bind *:3306
    mode tcp
    option tcpka # enables keep-alive both on client and server side
    balance roundrobin
    default-server port 9200
    server conn1 db4:13306 check inter 5s rise 1 fall 1 weight 3 maxconn 5000
    server conn2 db5:13306 check inter 5s rise 1 fall 1 weight 3 maxconn 5000
#---------------------------------------------------------------------
```

The hostname and port numbers should be modified to match your cluster configuration.

This solution will work for connection-based session IDs.

For correct operation within HAPerxy, a check script needs to be installed on all hosts running Tungsten Connector that will respond to a HAPerxy connector check script. It needs to be installed on all of the hosts running connectors and a xinet listener setup.

The connector check script will listen on port 9200 for connections from HAPerxy and will return the status of the connector to HAPerxy in the format of HTTP return codes.

To install the check script:

1. For the check to work, a mysql user must be created within the cluster which the check script can use. The user needs the permissions to be able to run the SQL in the check script:

   ```
   mysql> grant usage on *.* to haproxy identified by 'secret';
   ```

   If you are running smartscale the user will also need replication client privilege:

   ```
   mysql> grant usage, replication client on *.* to haproxy identified by 'secret';
   ```

2. Add the new user on each connector host by adding the following line to user.map.
haproxy secret cluster_name

3. Create and configure a check script on each host running Tungsten Connector. For example, create the file `/opt/continuent/share/connectorchk.sh`:

```bash
#!/bin/sh
#
# This script checks if a mysql server is healthy running on localhost. It will
# return:
# "HTTP/1.1 200 OK\r\n" (if mysql is running smoothly)
# - OR -
# "HTTP/1.1 503 Service Unavailable\r\n" (else)
#
# The purpose of this script is make haproxy capable of monitoring mysql properly

MYSQL_HOST=`hostname`
MYSQL_PORT="3306"            #Connector Port
MYSQL_USERNAME="haproxy"
MYSQL_PASSWORD="secret"
MYSQL_OPTS="-N -q -A test"

#If you create the following file, the proxy will return mysql down
#routing traffic to another host
FORCE_FAIL="/dev/shm/proxyoff"
OUT=""

return_ok() {
    echo -e "HTTP/1.1 200 OK\r\n"
    echo -e "Content-Type: text/plain\r\n"
    echo -e "MySQL is running.\r\n"
    echo -e "\r\n"
    exit 0
}

return_fail() {
    echo -e "HTTP/1.1 503 Service Unavailable\r\n"
    echo -e "Content-Type: text/plain\r\n"
    echo -e "MySQL is down.\r\n"
    echo -e "$OUT\r\n"
    exit 1
}

if [ -f "$FORCE_FAIL" ]; then
    OUT="$FORCE_FAIL found"
    return_fail;
fi

OUT=`mysql $MYSQL_OPTS --host=$MYSQL_HOST --port=$MYSQL_PORT --user=$MYSQL_USERNAME --password=$MYSQL_PASSWORD -e "select @@hostname;" 2>&1`
if [ $? -ne 0 ]; then
    return_fail;
fi
return_ok;
```

Set the permissions for the check script:

```bash
chown tungsten.tungsten /opt/continuent/share/connectorchk.sh
chmod 700 /opt/continuent/share/connectorchk.sh
chmod +x /opt/continuent/share/connectorchk.sh
```

4. Install `xinetd` and add the xinetd service. On RedHat/CentOS:

```bash
yum -y install xinetd telnet
```

On Debian/Ubuntu:

```bash
apt-get install xinetd telnet
```

5. Add an entry for the connector check script to `/etc/services`:

```bash
echo "connectorchk         9200/tcp" >> /etc/services
```

6. Add a configuration to `xinetd` by creating the file `/etc/xinetd.d/connectorchk` with the following content:

```bash
# default: on
# description: connectorchk
service connectorchk
{
    flags = REUSE
```
socket_type = stream
port = 9200
wait = no
user = tungsten
server = /opt/continuent/share/connectorchk.sh
log_on_failure => USERID
disable = no
# only_from = 0.0.0.0/0
# recommended to put the IPs that need
# to connect exclusively (security purposes)
per_source = UNLIMITED
}

7. Now restart **xinetd**:

```shell
$ service xinetd restart
```

8. Check the service is running:

```shell
$ telnet localhost 9200
```

You should get a response similar to this:

HTTP/1.1 200 OK
Content-Type: Content-Type: text/plain

MySQL is running.

### 5.7.8. Connection Pools

### 5.8. Connector Inline Command Interface

When connected to a service through Tungsten Connector, the connection has access to a number of specialized commands that can be executed.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tungsten cluster status</td>
<td>Displays a detailed view of the information the connector has about the cluster</td>
</tr>
<tr>
<td>tungsten connection count</td>
<td>Display the current number of active connection to each datasource</td>
</tr>
<tr>
<td>tungsten connection status</td>
<td>Displays information about the connection status for the last</td>
</tr>
<tr>
<td></td>
<td>statement executed</td>
</tr>
<tr>
<td>tungsten flush privileges</td>
<td>Reload the user.map file and update the user credentials</td>
</tr>
<tr>
<td>tungsten gc</td>
<td>Executes the connector garbage collector to free memory</td>
</tr>
<tr>
<td>tungsten help</td>
<td>Shows help description each statement</td>
</tr>
<tr>
<td>tungsten mem info</td>
<td>Display the memory usage information for the connector</td>
</tr>
<tr>
<td>tungsten show processlist</td>
<td>List all active queries on this connector instance</td>
</tr>
<tr>
<td>tungsten show variables</td>
<td>Display the connector configuration options currently in use</td>
</tr>
</tbody>
</table>

#### 5.8.1. tungsten cluster status

Shows the current cluster status, as far as the connector is aware. The output consists of a table showing dataservices and hosts and current status and role information:

```sql
tungsten cluster status;
```

<table>
<thead>
<tr>
<th>dataServiceName</th>
<th>name</th>
<th>host</th>
<th>role</th>
<th>state</th>
<th>appliedLatency</th>
<th>activeConnectionCount</th>
<th>connectionsCreatedCount</th>
</tr>
</thead>
<tbody>
<tr>
<td>alpha</td>
<td>host1</td>
<td>host1</td>
<td>slave</td>
<td>SHUNNED</td>
<td>60.0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>alpha</td>
<td>host2</td>
<td>host1</td>
<td>master</td>
<td>ONLINE</td>
<td>0.0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>alpha</td>
<td>host3</td>
<td>host3</td>
<td>slave</td>
<td>SHUNNED</td>
<td>61.0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

3 rows in set (0.01 sec)

The output fields are as follows:
• **dataServiceName**
  The name of the service. In connectors configured with multiple services, including composite clusters, there will be an entry for each host/servicename combination.

• **name**
  The name of the host within the service.

• **host**
  The hostname on which the service is running.

• **role**
  The current role for the host.

• **state**
  The current state for the host within the service.

• **appliedLatency**
  The applied latency of transactions; for masters the difference between the commit time and extraction, in a slave, the difference between commit time in the master and commit time in the slave.

• **activeConnectionCount**
  Count of the current number of active connections.

• **connectionsCreatedCount**
  Count of the number of connections created since the connector has been started.

### 5.8.2. tungsten connection count
Displays the current list of open connections for all hosts within the cluster.

```
mysql> tungsten connection count;
+--------------+-------------+----------------------------------------------------------------------+
| Data service | Data source | Active Connections                                                   |
+--------------+-------------+----------------------------------------------------------------------+
| alpha        | ct-multi1   | [internal(OPEN) DIRECT TO ct-multi1@alpha(master:ONLINE) STATUS(OK)] |
| alpha        | ct-multi2   | []                                                                    |
| alpha        | ct-multi3   | []                                                                    |
+--------------+-------------+----------------------------------------------------------------------+
3 rows in set (0.00 sec)
```

### 5.8.3. tungsten connection status
Displays the current connection status, for all connections, including indicating whether the connection is using SSL on either the incoming (client) or outgoing (MySQL) connection:

```
mysql> tungsten connection status;
+-------------------------------------------------------------------------------------+
| Message                                                                             |
+-------------------------------------------------------------------------------------+
| ct-multi1@alpha(master:ONLINE) STATUS(OK), QOS=RW_STRICT SSL.IN=false SSL.OUT=false |
+-------------------------------------------------------------------------------------+
1 row in set (0.05 sec)
```

### 5.8.4. tungsten flush privileges
Forces a reload of the user.map file to update the user privileges configured within the file. Only forces an update of the connector to which the client is connected.

```
mysql> tungsten flush privileges;
+--------------------------------+                             |
| Message                        |                             |
+--------------------------------+                             |
| user.map reloaded successfully |                             |
+--------------------------------+                             |
1 row in set (0.00 sec)
```
5.8.5. tungsten gc

Forces a Java garbage collection for the connector, recovering memory. An example of the memory used, garbage collection, and resulting memory usage below.

```
mysql> tungsten mem info;
+-----------------------+-------------------------------------------------------+
| JVM Memory statistics | Value in bytes                                        |
+-----------------------+-------------------------------------------------------+
| Peak Thread Count     | 18                                                    |
| Heap Memory           | init = 67108864(65536K) used = 17437496(17028K)       |
|                       | committed = 64886460(63360K) max = 259922560(253440K) |
|                       | committed = 24313856(23744K) used = 13970024(13642K) |
| Thread Count          | 16                                                    |
+-----------------------+-------------------------------------------------------+
4 rows in set (0.05 sec)

mysql> tungsten gc;
+-------------------------------+
| Message                       |
| Garbage collection successful |
+-------------------------------+
1 row in set (0.41 sec)

mysql> tungsten mem info;
+-----------------------+-------------------------------------------------------+
| JVM Memory statistics | Value in bytes                                        |
+-----------------------+-------------------------------------------------------+
| Peak Thread Count     | 18                                                    |
| Heap Memory           | init = 67108864(65536K) used = 4110088(4013K)         |
|                       | committed = 64946176(63424K) max = 259922560(253440K) |
|                       | committed = 24313856(23744K) used = 13970024(13642K) |
| Thread Count          | 16                                                    |
+-----------------------+-------------------------------------------------------+
4 rows in set (0.00 sec)
```

5.8.6. tungsten help

Displays the list of currently supported commands within the connector inline interface:

```
mysql> tungsten help;
+---------------------------------------------------------------------------+
| Message                                                                   |
| tungsten connection status:                 display information about the  |
|                                             connection used for the last   |
|                                             request ran                   |
| tungsten connection count:                  gives the count of current      |
|                                             connections to each one of the |
|                                             cluster datasources           |
| tungsten cluster status:                    prints detailed information    |
|                                             about the cluster view this   |
|                                             connector has                 |
| tungsten show [full] processlist:           list all running queries      |
|                                             handled by this connector     |
|                                             instance                      |
| tungsten show variables [like 'string']:    list connector configuration  |
|                                             options in use. The 'string'  |
|                                             may contain '%' wildcards     |
| tungsten flush privileges:                  reload user.map and refresh    |
|                                             user credentials              |
| tungsten mem info:                          display memory information    |
|                                             about current JVM            |
| tungsten gc:                                calls garbage collector       |
| tungsten help:                              display this help message     |
+---------------------------------------------------------------------------+
9 rows in set (0.00 sec)
```

5.8.7. tungsten mem info

```
mysql> tungsten mem info;
+-----------------------+-------------------------------------------------------+
| JVM Memory statistics | Value in bytes                                        |
+-----------------------+-------------------------------------------------------+
| Peak Thread Count     | 18                                                    |
| Heap Memory           | init = 67108864(65536K) used = 13469328(13153K)       |
|                       | committed = 64946176(63424K) max = 259922560(253440K) |
|                       | committed = 24313856(23744K) used = 13970024(13642K) |
+-----------------------+-------------------------------------------------------+
```

```
5.8.8. tungsten show [full] processlist

```
mysql> tungsten show processlist;
+------------+--------+----------+---------------+----------------+---------+------+-------+------+
| DataSource | Id     | User     | Host          | db             | Command | Time | State | Info |
|------------+--------+----------+---------------+----------------+---------+------+-------+------+
| host1      | 218886 | tungsten  | client1:57739 | tungsten_alpha  | Sleep   |  316 |       | NULL |
| host1      | 218925 | tungsten  | client2:58552 | tungsten_alpha  | Sleep   |  281 |       | NULL |
| host1      | 218932 | tungsten  | client1:57765 | tungsten_alpha  | Sleep   |  274 |       | NULL |
+------------+--------+----------+---------------+----------------+---------+------+-------+------+
```

5.8.9. show slave status

The `SHOW SLAVE STATUS` command generates a version of the standard MySQL command, with the output replaced with values generated by Tungsten. This can be useful for environments where the slave status need to be checked, but with the Tungsten Replicator state, rather than native replication.

```
mysql> tungsten show slave status;
*************************** 1. row ***************************
Slave_IO_State:         
Master_Host: host1      
Master_User:             
Master_Port: 0           
Connect_Retry: 0         
Master_Log_File: mysql-bin.mysql-bin.000050 
Read_Master_Log_Pos: 0   
Relay_Master_Log_File:   
Relay_Log_File:          
Relay_Log_Pos: 0         
Relay_Master_Log_File:   
Slave_IO_Running:        
Slave_SQL_Running:       
Replicate_Do_DB:         
Replicate_Ignore_DB:     
Replicate_Do_Table:      
Replicate_Ignore_Table:  
Replicate_Wild_Do_Table: 
Replicate_Wild_Ignore_Table:
Last_Errno: 0            
Last_Error:              
Skip_Counter: 0          
Exec_Master_Log_Pos: 1269
Relay_Log_Space: 0       
Until_Condition:         
Until_Log_File:          
Until_Log_Pos: 0         
Master_SSL_Allowed:      
Master_SSL_CA_File:      
Master_SSL_CA_Path:      
Master_SSL_Cert:         
Master_SSL_CIPHER:       
Master_SSL_Verify_Server_Cert:
Seconds_Behind_Master: 0 
Master_SSL_VERIFY_SERVER_Cert:
Last_IO_Error: 0         
Last_SQL_Error: 0        
Last_SQL_Error: 0        
Last_SQL_Error: 0        
1 row in set (0.01 sec)
```

5.8.10. tungsten show variables

```
mysql> tungsten show variables;
+---------------------------------+--------------------------------------------------------------+
| Variable_name                   | Value                                                        |
|---------------------------------+--------------------------------------------------------------+
| AuthorizedHostsFileName         | ../conf/authorized_hosts                                     |
| AutoReconnect                   | true                                                         |
| BridgeClientToServerBufferSize  | 1024                                                         |
| BridgeMode                      | OFF                                                          |
| BridgeServerToClientBufferSize  | 1024                                                         |
| ConnectionCloseIdleTimeout      | 0                                                            |
| DebugMode                       | false                                                        |
| DirectReads                     | false                                                        |
```
5.9. Advanced Configuration

5.9.1. Using SSL Connections

5.9.2. Changing the Connector Listen Address or Interface

5.9.3. Connector Performance

5.9.4. Using Multiple Dataservices

5.10. Connector General Limitations

The following general limitations exist when using the Connector; these issues do not affect the connector when using it in when using bridge mode:

- When using `mysqldump` within MySQL 5.6 or later, the `--single-transaction` option is not supported when connectivity to the database is made through the connector.

- When using `mysqldump`, the `--flush-logs` option is not supported when connectivity to the database is made through the connector.
Chapter 6. Tungsten Manager

The Tungsten Manager provides the management and monitoring of the Continuent Tungsten services to ensure that datasources, connectors and other components are running, datasources are replicating to each other, and handles failover and maintenance schedules.

6.1. Manager API

Table 6.1. Manager REST API Endpoints

<table>
<thead>
<tr>
<th>HTTP Method</th>
<th>URI</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/ping</td>
<td></td>
</tr>
<tr>
<td>GET</td>
<td>/status</td>
<td></td>
</tr>
</tbody>
</table>

| Description                                           | Status information on dataService and compositeDataService                  |
| Method       | GET /ping        |                                                                             |
| Request Data | None             |                                                                             |
| Response Data| PingDescriptor   |                                                                             |
| Authentication Required | no           |                                                                             |

| Return Codes | 200 | Request completed successfully. |

| Description                                           | Get dataService status                                                     |
| Method       | GET /status/{service} |                                                                             |
| Request Data | None             |                                                                             |
| Response Data| ClusterApiResponse .getOutputPayload().getDataServiceState() |                                                                             |
| Authentication Required | no           |                                                                             |
| Path Arguments| service | dataService or compositeDataService name                                    |
| Parameters   | string; required |                                                                             |

| Return Codes | 200 | Request completed successfully. |
Chapter 7. Command-line Tools

Continuent Tungsten 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 Continuent Tungsten 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, although all deployments rely on the tpm command.

Commands related to the deployment
- **tpm** — Tungsten package manager

Commands related to managing Continuent Tungsten
- **cctrl** — cluster control
- **cluster_backup** — cluster backup automation

Commands related to the core Tungsten Replicator
- **trepctl** — replicator control
- **multi_trepctl** — multi-replicator control
- **thl** — examine Tungsten History Log contents

Commands related to managing Tungsten Replicator deployments
- **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 monitoring the cluster service
- **tungsten_monitor** — build DDL, materialize and compare replicated data
- **tungsten_health_check** — checks the cluster for best practice configuration and operation

### 7.1. 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>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-c</td>
<td>Check the Connector service status.</td>
</tr>
<tr>
<td>-h</td>
<td>Display the help text.</td>
</tr>
<tr>
<td>-r</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:
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.

**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
  ```

- To check the replicator and manager services are executing:
  ```shell
  check_tungsten_services -r
  OK: All services (Replicator, Manager) are running
  ```

- To check the connector services:
  ```shell
  check_tungsten_services -c
  OK: All services (Replicator) are online
  ```

### 7.2. The `check_tungsten_online` Command

The `check_tungsten_online` command checks whether all the hosts in a given service are online and running.

Within a Continuent Tungsten service, the replicator, manager and connector services are checked. All must be online for an OK response.

**Table 7.2. `check_tungsten_online` Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-s SERVICENAME</code></td>
<td>Check which hosts within a given service and online.</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:
  
  ```
  CRITICAL: Replicator is not running
  ```

- **WARNING** — status requires attention. This indicates that one service within the system is not online.

- **OK** — status is OK.
  
  For example:
  
  ```
  OK: All services are online
  ```

This output is easily parseable by various monitoring tools, including Nagios NRPE, and can be used to monitor the status of your services quickly without resorting to using the full `trepctl` output.

For example:

```shell
check_tungsten_online
OK: All services are online
```

If you have multiple services installed, use the `-s` to specify the service:

```shell
check_tungsten_online -s alpha
OK: All services are online
```
7.3. 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 7.3. check_tungsten_latency Options**

<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>

The command outputs information in the following format:

```
LEVEL: DETAIL
```

Where DETAIL includes detailed information about the status report, and LEVEL is:

- **CRITICAL** — latency on at least one node is above the specified threshold level for a critical report. The host reporting the high latency will be included in the DETAIL portion:

  For example:
  ```
  CRITICAL: host2=0.506s
  ```

- **WARNING** — latency on at least one node is above the specified threshold level for a warning report. The host reporting the high latency will be included in the DETAIL portion:

  For example:
  ```
  WARNING: host2=0.506s
  ```

- **OK** — status is OK; the highest reported latency will be included in the output.

  For example:
  ```
  OK: All slaves are running normally (max_latency=0.506)
  ```

The `-w` and `-c` options must be specified on the command line, and the critical figure must be higher than the warning figure. For example:

```
shell> check_tungsten_latency -w 0.1 -c 0.5
CRITICAL: host2=0.506s
```

Performance information can be included in the output to monitor the status. The format for the output is included in the DETAIL block and separates the maximum latency information for each node with a semicolon, and the detail block with a pipe symbol. For example:

```
shell> check_tungsten_latency -w 1 -c 1 --perfdata
OK: All slaves are running normally (max_latency=0.506)
```

Performance information for all the slaves in the cluster can be output by using the `--perslave-perfdata` option which must be used in conjunction with the `--perfdata` option:

```
shell> check_tungsten_latency -w 0.2 -c 0.5 --perfdata --perslave-perfdata
CRITICAL: host2=0.506s | host1=0.0;0.2;0.5; host2=0.506;0.2;0.5;
```

7.4. The check_tungsten_progress Command

The `check_tungsten_progress` command determines whether the replicator is actually making progress by executing a heartbeat operation and monitoring for this operation to complete within an optional time period (default is 1 second).

**Table 7.4. check_tungsten_progress Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-t #</code></td>
<td>Give a time period during which progress should be identified</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:
Command-line Tools

- CRITICAL — replicator is not making progress and either has not completed the heartbeat operation, or has failed. If failed, the reason will be shown in the DETAIL:

  For example:

  ```
  CRITICAL: Replicator is not ONLINE
  ```

- OK — replicator is making progress.

  For example:

  ```
  OK: Replicator is making progress
  ```

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.

The time delay can be added on busy systems to ensure that the replicator is progressing; for example, to wait 15 seconds to ensure the replicator is progressing:

```shell
check_tungsten_progress -t 15
CRITICAL: Replicator is not ONLINE
```

### 7.5. The `cctrl` Command

The `cctrl` command provides cluster management for your installed cluster, providing a command-line shell interface to obtain information and manage your cluster and structure.

#### 7.5.1. `cctrl` Command-line Options

```
```

Where:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-admin</code></td>
<td>Enter admin mode when connecting</td>
</tr>
<tr>
<td><code>-expert</code></td>
<td>Enter expert mode when connecting</td>
</tr>
<tr>
<td><code>-host host</code></td>
<td>Host name of the service manager to use</td>
</tr>
<tr>
<td><code>-logical</code></td>
<td>Enter logical mode when connecting</td>
</tr>
<tr>
<td><code>-multi</code></td>
<td>Connector to multiple services</td>
</tr>
<tr>
<td><code>-no-history</code></td>
<td>Disable command history</td>
</tr>
<tr>
<td><code>-physical</code></td>
<td>Enter physical mode when connecting</td>
</tr>
<tr>
<td><code>-port port</code></td>
<td>Specify the TCP/IP port of the service manager</td>
</tr>
<tr>
<td><code>-proxy</code></td>
<td>Operate as a proxy service</td>
</tr>
<tr>
<td><code>-service</code></td>
<td>Connect to a specific service</td>
</tr>
</tbody>
</table>

- `-admin`
- `-expert`
- `-host`
- `-logical`
- `-multi`
- `-no-history`
- `-physical`
- `-port`
- `-proxy`
- `-service`
7.5.2. `cctrl` Modes

- Admin Mode
- Expert Mode
- Logical Mode
- Physical Mode

You can specify the mode to enter from the command-line, using the appropriate switch. For example, to start `cctrl` in Expert mode:

```
shell> cctrl -expert
```

The default mode is Logical.

You can also change the mode from within `cctrl` by issuing the appropriate command. For example, to switch to Expert mode:

```
[LOGICAL] /alpha > expert
```

**WARNING:** This is an expert-level command: Incorrect use may cause data corruption or make the cluster unavailable.

Do you want to continue? [y/n] > y

```
[LOGICAL:EXPERT] /alpha >
```

The current mode is always displayed as part of the command prompt within `cctrl`.

7.5.3. `cctrl` Commands

Table 7.6. `cctrl` Commands

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>admin</code></td>
<td>Change to admin mode</td>
</tr>
<tr>
<td><code>cd</code></td>
<td>Change to a specific site within a multisite service</td>
</tr>
<tr>
<td><code>cluster</code></td>
<td>Issue a command across the entire cluster</td>
</tr>
<tr>
<td><code>cluster validate</code></td>
<td>Validate the cluster quorum configuration</td>
</tr>
<tr>
<td><code>create composite</code></td>
<td>Create a composite dataservice</td>
</tr>
<tr>
<td><code>datasource</code></td>
<td>Issue a command on a single datasource</td>
</tr>
<tr>
<td><code>expert</code></td>
<td>Change to expert mode</td>
</tr>
<tr>
<td><code>failover</code></td>
<td>Perform a failover operation from a master to a slave</td>
</tr>
<tr>
<td><code>help</code></td>
<td>Display the help information</td>
</tr>
<tr>
<td><code>is</code></td>
<td>Show cluster status</td>
</tr>
<tr>
<td><code>members</code></td>
<td>List the managers of the dataservice</td>
</tr>
<tr>
<td><code>physical</code></td>
<td>Enter physical mode</td>
</tr>
<tr>
<td><code>ping</code></td>
<td>Test host availability</td>
</tr>
<tr>
<td><code>quit, exit</code></td>
<td>Exit <code>cctrl</code></td>
</tr>
<tr>
<td><code>recover master using</code></td>
<td>Recover the master within a datasource using the specified master</td>
</tr>
<tr>
<td><code>replicator</code></td>
<td>Issue a command on a specific replicator</td>
</tr>
<tr>
<td><code>router</code></td>
<td>Issue a command on a specific router (connector)</td>
</tr>
<tr>
<td><code>service</code></td>
<td>Run a service script</td>
</tr>
<tr>
<td><code>set</code></td>
<td>Set management options</td>
</tr>
<tr>
<td><code>set master</code></td>
<td>Set the master within a datasource</td>
</tr>
<tr>
<td><code>switch</code></td>
<td>Promote a slave to a master</td>
</tr>
</tbody>
</table>

7.5.3.1. `cctrl admin>` Command

The `admin` command enables admin mode commands and displays. Admin mode is a specialized mode used to examine and repair cluster metadata. It is not recommended for normal use.
7.5.3.2. cctrl cd Command

The cd command changes the data service being administered. Subsequent commands will only affect the given data service name. 'cd ..' allows to go back to the root element. The given data service name can be either composite or physical. Note that this command can only be used when cctrl is run with the '-multi' flag.

7.5.3.3. cctrl cluster Command

The cluster command operates at the level of the full cluster.

7.5.3.3.1. cctrl cluster check Command

The cluster check command issues an MD5 consistency check on one or more tables in a database on the master data source. The consistency checks then replicate to each slave, whereupon the slave replicator repeats the check.

If the check fails, slaves may go offline or print a log warning depending on how the replicators are configured. The default is to go offline. You can return a replicator to the online state after a failed check by issuing a replicator online command.

The table name can also be a wildcard (*) in which case all tables will be checked. Users may optionally specify a range of rows to check using the -limit option, which takes a starting row option followed by a number of rows to check. Rows are selected in primary key order.

Usage:
The following example checks all tables in database accounting.

[LOGICAL] /alpha > cluster check accounting.*

The following command checks only the first 10 rows in a single table.

[LOGICAL] /alpha > cluster check accounting.invoices -limit 1,10

Warning
Consistency checks can be very lengthy operations for large tables and will lock them while they run. On the master this can block applications. On slaves it blocks replication.

7.5.3.3.2. cctrl cluster flush Command

The cluster flush command sends a heartbeat event through the local cluster and returns a flush sequence number that is guaranteed to be equal to or greater than the sequence number of the flush event. Slaves that reach the flush sequence number are guaranteed to have applied the flush event.

This command is commonly used for operations like switch that need to synchronize the position of one or more masters or slaves.

7.5.3.3.3. cctrl cluster heartbeat Command

The cluster heartbeat command sends a heartbeat event through the local cluster to demonstrate that all replicators are working. You should see the sequence numbers on all data sources advance by at least 1 if it is successful.

7.5.3.3.4. cctrl cluster offline Command

The cluster offline command brings all data services that are not offline into the offline state. It has no effect on services that are already offline.

7.5.3.3.5. cctrl cluster online Command

The cluster online command brings all data services that are not online into the online state. It has no effect on services that are already online.

7.5.3.3.6. cctrl cluster validate Command

The cluster validate command validates the configuration of the cluster with respect to the quorum used for decision making. The number of active managers, active witnesses and passive witnesses within the cluster is validated to ensure that there are enough active hosts to make a decision in the event of a failover or other failure event.

When executed, the validation routine checks all the available hosts, including witness hosts, and determines whether there are enough hosts, and whether their membership of the cluster is valid. In the event of deficiency, corrective action will be recommended.

By default, the command checks all hosts within the configured cluster:

[LOGICAL] /alpha > cluster validate
HOST host1/192.168.2.20: ALIVE
HOST host2/192.168.2.21: ALIVE
HOST host3/192.168.2.22: ALIVE
CHECKING FOR QUORUM: MUST BE AT LEAST 2 MEMBERS, OR 1 MEMBERS PLUS ALL WITNESSES
QUORUM SET MEMBERS ARE: host2, host1, host3
SIMPLE MAJORITY SIZE: 2
VALIDATED MEMBERS ARE: host2, host1, host3
REACHABLE MEMBERS ARE: host2, host1, host3
WITNESS HOSTS ARE:
REACHABLE WITNESSES ARE:
MEMBERSHIP IS VALID
GC VIEW OF CURRENT MEMBERS IS: host1, host2, host3
VALIDATED CURRENT MEMBERS ARE: host2, host1, host3
CONCLUSION: I AM IN A PRIMARY PARTITION OF 3 MEMBERS OUT OF THE REQUIRED MAJORITY OF 2
VALIDATION STATUS=VALID CLUSTER
ACTION=NONE

Additionally, a list of hosts to exclude from the check can be provided to verify the cluster capability when certain hosts have already failed or been shunned from the dataservice during maintenance.

To exclude hosts, add excluding and a comma-separated list of hosts to the command. For example:

```
[LOGICAL] /alpha > cluster validate excluding host3,host2
EXCLUDED host3 FROM VIEW
EXCLUDED host2 FROM VIEW
HOST host1/192.168.2.20: ALIVE
CHECKING FOR QUORUM: MUST BE AT LEAST 2 MEMBERS, OR 1 MEMBERS PLUS ALL WITNESSES
QUORUM SET MEMBERS ARE: host2, host1, host3
SIMPLE MAJORITY SIZE: 2
VALIDATED MEMBERS ARE: host1
REACHABLE MEMBERS ARE: host1
WITNESS HOSTS ARE:
REACHABLE WITNESSES ARE:
MEMBERSHIP IS VALID
GC VIEW OF CURRENT MEMBERS IS: host1
VALIDATED CURRENT MEMBERS ARE: host1
CONCLUSION: I AM IN A NON-PRIMARY PARTITION OF 1 MEMBERS OUT OF A REQUIRED MAJORITY SIZE OF 2
AND THERE ARE 0 REACHABLE WITNESSES OUT OF 0
VALIDATION STATUS=NON-PRIMARY PARTITION
ACTION=RESTART SAFE
```

Cluster validation can be used to provide validation only. To improve the support:

- Add active witnesses to the dataservice, see Section 2.9.2, “Adding Active Witnesses to an Existing Deployment”
- Add slave hosts to the dataservice, see Section 2.9.1, “Adding Datasources to an Existing Deployment”
- Add passive witnesses to the dataservice, see Section 2.9.3, “Adding Passive Witnesses to an Existing Deployment”

### 7.5.3.4. `cctrl create composite` Command

The `create composite` command creates a new composite data source or data service with the given name. Composite data services can only be create in the root directory ‘/’ while composite data sources need to be created from a composite data service location. Composite data source names should be the same as the physical data services Composite data service name should be named after its composite data sources

Usage:

The following example creates a composite data service named ‘sj_nyc’

```
create composite dataservice sj_nyc
```

The following example changes to the composite data service sj_nyc, then creates a composite data source named ‘sj’ in this composite data service

```
create composite datasource sj
```

### 7.5.3.5. `cctrl datasource` Command

The datasource command affects a single data source.

```
datasource
fail
host
offline
online
```
recover
restore
shun
welcome

Table 7.7. ccctrl datasource Commands

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fail</td>
<td>Fail a datasource</td>
</tr>
<tr>
<td>host</td>
<td>Hostname of the datasource</td>
</tr>
<tr>
<td>offline</td>
<td>Put a datasource into the offline state</td>
</tr>
<tr>
<td>online</td>
<td>Put a datasource into the online state</td>
</tr>
<tr>
<td>recover</td>
<td>Recover a datasource into operation state as slave</td>
</tr>
<tr>
<td>restore</td>
<td>Restore a datasource from a previous backup</td>
</tr>
<tr>
<td>shun</td>
<td>Shun a datasource</td>
</tr>
<tr>
<td>welcome</td>
<td>Welcome a shunned datasource back to the cluster</td>
</tr>
</tbody>
</table>

7.5.3.5.1. ccctrl datasource backup Command

The datasource backup command invokes a backup on the data source on the named host using the default backup agent and storage agent. Backups taken in this way can be reloaded using the ‘datasource restore’ command. The following command options are supported:

- `backupAgent` - The name of a backup agent.
- `storageAgent` - The name of a storage agent.
- `timeout` - Number of seconds to wait before the backup command times out.

On success the backup URL will be written to the console.

Usage:

The following example performs a backup on host saturn using the default backup agent.

```
cctrl> datasource saturn backup
```

The following example performs a backup on host mercury using the xtrabackup agent, which is named explicitly.

```
cctrl> datasource mercury backup xtrabackup
```

7.5.3.5.2. ccctrl datasource backup Command

7.5.3.5.3. ccctrl datasource host Command

7.5.3.5.4. ccctrl datasource offline Command

7.5.3.5.5. ccctrl datasource online Command

7.5.3.5.6. ccctrl datasource recover Command

7.5.3.5.7. ccctrl datasource restore Command

7.5.3.5.8. ccctrl datasource shun Command

7.5.3.5.9. ccctrl datasource welcome Command

When a datasource has been shunned, the datasource can be welcomed back to the dataservice by using the `welcome` command. The `welcome` command attempts to enable the datasource in the ONLINE state using the current roles and configuration. If the datasource was operating as a slave before it was shunned, the `welcome` command will enable the datasource as a slave.

For example, the host `host3` is a slave and currently online:
To switch the datasource back to the online state, the `welcome` is used:

```
[LOGICAL:EXPERT] /alpha > datasource host3 welcome
Data Source 'host3' is now OFFLINE
```

The `welcome` command puts the datasource into the Offline state. If the dataservice policy mode is Automatic, the node will be placed into Online mode due to automatic recovery. When in Maintenance or Manual mode, the node must be manually set online.

The `welcome` command may not always work if there has been a failure or topology change between the moment it was shunned and welcomed back. Using the `recover` command may be a better alternative to using `welcome` when bringing a datasource back online. The `recover` commands ensures that the replicator, connector and operation of the datasource are correct within the current cluster configuration. See Section 7.5.3.14, “cctrl recover Command”.

### 7.5.3.6. cctrl expert Command

### 7.5.3.7. cctrl failover Command

### 7.5.3.8. cctrl help Command

The `help` command provides help text from within the `cctrl` operation.

With no other arguments, `help` provides a list of the available commands:

```
[LOGICAL] /alpha &gt; &lt;userinput&gt;help&lt;/userinput&gt;
--------
Overview
--------
Description: Overview of Tungsten cctrl Commands
Commands
--------
admin                          - Enter admin mode
cd &lt;name&gt;                      - Change to the specified SOR cluster element
cluster &lt;command&gt;              - Issue a command on the entire cluster
create composite &lt;type&gt; &lt;name&gt;  - Create SOR cluster components
datasource &lt;host&gt; &lt;cmd&gt;        - Issue a command on a datasource
expert                          - Enter expert mode
failover                       - Failover from failed master to slave
help                           - Show help
is [options]                   - Show generic cluster status
members                        - List all of the managers in the cluster
ping                           - Test host availability
physical                      - Enter physical mode
quit or exit                   - Leave cctrl
replicator &lt;host&gt; &lt;cmd&gt;        - Issue a command on a replicator
service                        - Run a service script
set                            - Set management options
switch                         - Promote a slave to master

To get more information about particular commands type help followed by a command. Examples: 'help datasource' or 'help create composite'.

To get specific information about an individual command or operation, provide the command name to the `help` command. For example, to get information about the `ping` command, type `help ping` at the `cctrl` prompt.

### 7.5.3.9. cctrl ls Command

The `ls` command displays the current structure and status of the cluster.

```
ls [-l] [host] [[resources] | [services] | [sessions]]
```

The `ls` command operates in a number of different modes, according to the options provided on the command-line, as follows:
• No options

Generates a list of the current routers, datasources, and their current status and services.

• `-l`

Outputs extended information about the current status and configuration. The `-l` option can be used in both the standard (no option) and host specific output formats to provide more detailed information.

• `host`

• `resources`

• `services`

• `sessions`

Without any further options, the

```
[LOGICAL] /alpha > <userinput>ls</userinput>
COORDINATOR[host1:AUTOMATIC:ONLINE]
ROUTERS:
  connector@host1[1179](ONLINE, created=0, active=0)
  connector@host2[1532](ONLINE, created=0, active=0)
  connector@host3[17665](ONLINE, created=0, active=0)
```

```
DATASOURCES:
  host1(master:ONLINE, progress=60, THL latency=0.498)
  STATUS [OK] [2013/03/22 02:25:00 PM GMT]
  MANAGER(state=ONLINE)
  REPLICATOR(role=master, state=ONLINE)
  DATASERVER(state=ONLINE)
  CONNECTIONS(created=0, active=0)
```

```
  host2(slave:ONLINE, progress=31, latency=0.000)
  STATUS [OK] [2013/03/22 02:25:00 PM GMT]
  MANAGER(state=ONLINE)
  REPLICATOR(role=slave, master=host1, state=ONLINE)
  DATASERVER(state=ONLINE)
  CONNECTIONS(created=0, active=0)
```

```
  host3(slave:ONLINE, progress=35, latency=9.455)
  STATUS [OK] [2013/03/21 06:47:53 PM GMT]
  MANAGER(state=ONLINE)
  REPLICATOR(role=slave, master=host1, state=ONLINE)
  DATASERVER(state=ONLINE)
  CONNECTIONS(created=0, active=0)
```

7.5.3.9.1. cctrl ls host

You can also specify an individual component within the cluster on which to obtain information. For example, to get the information only for a single host:

```
[LOGICAL] /alpha > <userinput>ls host2</userinput>
COORDINATOR[host1:AUTOMATIC:ONLINE]
ROUTERS:
  connector@host1[1179](ONLINE, created=0, active=0)
  connector@host2[1532](ONLINE, created=0, active=0)
  connector@host3[17665](ONLINE, created=0, active=0)
```
7.5.3.9.2. cctrl ls -l (Extended Information)

7.5.3.9.3. cctrl ls resources

The resources option generates a list of the configured resources and their current status.

7.5.3.9.4. cctrl ls services

7.5.3.9.5. cctrl ls sessions

7.5.3.10. cctrl members Command

The members command outputs a list of the currently identified managers within the dataservice.

members

For example:

[LOGICAL]/alpha > <userinput>members</userinput>
alpha/host1(ONLINE)/192.168.1.60:7800
alpha/host2(ONLINE)/192.168.1.61:7800
alpha/host3(ONLINE)/192.168.1.62:7800

The command outputs each identified manager service within the current dataservice.

The format of the output information is:

DATASERVICE/HOST (STATUS)/IPADDR:PORT

Where:

• DATASERVICE
  The name of the dataservice.

• HOST
  The name of the host on which the manager resides.

• STATUS
  The current status of the manager.

• IPADDR
  The IP address of the manager.

• PORT
  The primary TCP/IP port used for contacting the manager service.

The members service can be used as an indicator of the overall status of the dataservice. The information shown for each manager should within a single dataservice should be identical. If different information is shown, or an incomplete number of managers compared to the number of configured managers is provided, then it may indicate a communication or partition problem within the dataservice.

7.5.3.11. cctrl physical Command

7.5.3.12. cctrl ping Command
7.5.3.13. cctrl quit Command

Exits cctrl and returns the user to the shell. For example:

7.5.3.14. cctrl recover Command

7.5.3.15. cctrl recover master using Command

7.5.3.16. cctrl recover relay using Command

7.5.3.17. cctrl replicator Command

7.5.3.18. cctrl router Command

7.5.3.19. cctrl service Command

7.5.3.20. cctrl set Command

7.5.3.21. cctrl set master Command

7.5.3.22. cctrl switch Command

7.6. The cluster_backup Command

The cluster_backup command provides a simple mechanism to execute a Tungsten Replicator backup inside of the cluster. It is designed to be called manually or as part of cron. The command should be added to cron on every server. When started, the command will check if the server is the current coordinator for the cluster. If not, the command will exit without an error. This design ensures that the command will only run on one server in the cluster.

The command supports command-line options that allow you to alter how and where the backup is executed.

Table 7.8. cluster_backup Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--require-automatic-mode</td>
<td>Require that the cluster be in AUTOMATIC mode before taking a backup. Default: true</td>
</tr>
<tr>
<td>--require-slave-backup</td>
<td>Require that the backup is taken of a slave datasource. If this is enabled and there are no slaves available, no backup will be taken. Default: true</td>
</tr>
<tr>
<td>--offline-backup</td>
<td>Put the datasource OFFLINE prior to taking the backup. This should not be enabled if you disable --require-slave-backup. Default: false</td>
</tr>
<tr>
<td>--agent String</td>
<td>The Tungsten Replicator backup agent to use when taking the backup. If no value is given the default backup agent will be used.</td>
</tr>
<tr>
<td>--datasource String</td>
<td>Execute the backup against this datasource.</td>
</tr>
</tbody>
</table>

After the command confirms the current server is the coordinator, it will attempt to find a datasource to backup. Unless --require-slave-backup has been disabled, only slaves that are ONLINE will be eligible. If no datasource can be found, the command will exit with an error. The backup will then be started on the datasource.

The cluster_backup command will wait until the cctrl command has returned before exiting. The cctrl command can return prior to the backup is completed if it takes too long or if there is another error. The tungsten_nagios_backups check or similar should be used to make sure that you always have a recent backup available in the cluster.

7.7. 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 7.9, "thl Options".

Table 7.9. thl Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-conf path</td>
<td>Path to the configuration file containing the required replicator service configuration</td>
</tr>
<tr>
<td>-service servicename</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 parameter 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.

### 7.7.1. thl list Command

The list parameter to the thl command outputs a list of the sequence number information from the THL. By default, the entire THL as stored on disk is output. Command-line options enable you to select individual sequence numbers, sequence number ranges, or all the sequence information from a single file.

```
```

There are three selection mechanisms:

- **-seqno #**

Output the THL sequence for the specific sequence number. When reviewing or searching for a specific sequence number, for example when the application of a sequence on a 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)
- TIME = 2013-05-02 11:37:00.0
- EPOCH# = 7
- EVENTID = mysql-bin.000004:0000000000003345;0
- SOURCEID = host1
- METADATA = [mysql_server_id=1687011;unsafe_for_block_commit;dbms_type=mysql;service=firstrep;shard=cheffy]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [##charset = UTF-8, autocommit = 1, sql_auto_is_null = 0, foreign_key_checks = 0, unique_checks = 0, sql_mode = 'NO_AUTO_VALUE_ON_ZERO', character_set_client = 33, collation_connection = 33, collation_server = 8]
- SCHEMA = cheffy
- SQL(0) = CREATE TABLE 'access_log' ('
  'id' int(10) unsigned NOT NULL AUTO_INCREMENT,'
  'userid' int(10) unsigned DEFAULT NULL,'
  'datetime' int(10) unsigned NOT NULL DEFAULT '0',
  ...
```

If the sequence number selected contains multiple fragments, each fragment will be output. Depending on the content of the sequence number information, the information can be output containing only the header/metadata information or only the table data (row or SQL) that was contained within the fragment. See **-headers** and **-sql** for more information.

**Note**

Unsigned integers are displayed and stored in the THL as their negative equivalents, and translated to the correct unsigned type when the data is applied to the target database.
-low # and/or -high #

Specify the start (-low [166]) or end (-high [166]) of the range of sequence numbers to be output. If only -low [166] is specified, then all sequence numbers from that number to the end of the THL are output. If -high [166] is specified, all sequence numbers from the start of the available log file to the specified sequence number are output. If both numbers are specified, output all the sequence numbers within the specified range. For example:

```
shell> thl list -low 320
```

Will output all the sequence number fragments from number 320.

```
shell> thl list -high 540
```

Will output all the sequence number fragments up to and including 540.

```
shell> thl list -low 320 -high 540
```

Will output all the sequence number fragments from number 320 up to, and including, sequence number 540.

- file filename [166]

Outputs all of the sequence number fragment information from the specified THL file. If the filename has been determined from the thl index command, or by examining the output of other fragments, the file-based output can be used to identify statements or row data within the THL.

- charset charset [166]

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 [166]

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 [166]

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.000005:0000000580759206:0 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 D.1.1, "THL Format".

- json

Only valid with the -headers option, the header information is output for the selected sequence numbers from the THL in JSON format. The field contents are identical, with each fragment of each THL sequence being contained in a JSON object, with the output consisting of an array of the these sequence objects. For example:

```
[
]
```
FOR More information on the fields displayed, see THL SEQNO [359].

- -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=1687011;dbms_type=mysql;is_metadata=true;service=firstrep;shard=tungsten_firstrep;heartbeat=MASTER_ONLINE]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- SQL(0) =
- ACTION = UPDATE
- SCHEMA = tungsten_firstrep
- TABLE = heartbeat
- ROW# = 0
- COL(index=1 name= type=4 [INTEGER] length=8 unsigned=false blob=false desc=null) = 1
- COL(index=2 name= type=4 [INTEGER] length=8 unsigned=false blob=false desc=null) = 1416
- COL(index=3 name= type=12 [VARCHAR] length=0 unsigned=false blob=false desc=null) = [B@65b60280
- COL(index=4 name= type=93 [TIMESTAMP] length=0 unsigned=false blob=false desc=null) = 2014-01-30 05:46:26.0
- COL(index=5 name= type=93 [TIMESTAMP] length=0 unsigned=false blob=false desc=null) = 2013-05-03 12:05:47.0
- COL(index=6 name= type=4 [INTEGER] length=8 unsigned=false blob=false desc=null) = 1015
- COL(index=7 name= type=4 [INTEGER] length=8 unsigned=false blob=false desc=null) = 0
- COL(index=8 name= type=12 [VARCHAR] length=0 unsigned=false blob=false desc=null) = [B@105e55ab
- KEY(index=1 name= type=4 [INTEGER] length=8 unsigned=false blob=false desc=null) = 1
```

When identifying the different data types, the following effects should be noted:

- **CHAR** and **VARCHAR** are identified as type 12, **VARCHAR**
- **SET** is identified as an **INTEGER**
- When the value is either **NULL** or 0 (Zero), date and time fields are shown as type 0, **NULL**
- **ENUM** is identified as an **OTHER**
- **BLOB** and **TEXT** are identified as type 2004, **BLOB**

- **--timezone**

Specify the timezone to use when display date or time values. When not specified, times are displayed using UTC.

### 7.7.2. thl index Command

The **index** parameter to **thl** provides a list of all the available THL files and the sequence number range stored within each file:

```
shell> thl index
LogIndexEntry thl.data.0000000001(0:113)
LogIndexEntry thl.data.0000000002(114:278)
LogIndexEntry thl.data.0000000003(279:375)
LogIndexEntry thl.data.0000000004(376:472)
```
7.7.3. thl purge Command

The optional argument `-no-checksum` ignores the checksum information on events in the event that the checksum is corrupt.

The `thl purge` command deletes sequence number information from the THL files.

```
thl purge [-low #] [-high #] [-y] [-no-checksum]
```

The `purge` parameter deletes the THL data according to the following rules:

- Without any specification, a `purge` command will delete all of the stored THL information.
- 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` parameter, 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, the command below deletes all entries up to and included 3670:

```
shell> thl purge -high 3670
```

The warning message can be ignored by using the `-y` option, which implies that the operation should proceed without further confirmation.

The optional argument `-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.
- 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.

7.7.4. thl info Command

The optional argument `-no-checksum` ignores the checksum information on events in the event that the checksum is corrupt.

7.7.5. thl help Command

The `help` parameter to the `thl` command outputs the current help message text.

7.8. 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 7.8.1, “trepctl Options”.

```
trepctl
  backup [-backup agent] [-limit s] [-storage agent]
capabilities
  check
clear
  clients [-json]
  flush [-limit s]
  heartbeat [-name] [-host name]
  kill [-y]
  load
  offline
  offline-deferred [-at-event event] [-at-heartbeat [heartbeat]] [-at-seqno seqno] [-at-time YYYY-MM-DD hh:mm:ss] [-immediate]
  properties [-filter name]
  purge [-limit s]
  reset [-y]
  restore [-retry N] [-service name]
  services [-full] [-json]
  setrole [-role master relay slave] [-uri]
  shard [-delete shard] [-insert shard] [-list] [-update shard]
  shutdown [-y]
  start
  status [-json] [-namechannel-assignments servicesshardsstagesstoreswatchswatches]
  stop [-y]
  unload [-verbose]
  version
  wait [-applied seqno] [-limit s] [-state st]
```

For individual operations, `trepctl` uses a sub-command structure on the command-line that specifies which operation is to be performed. There are two classifications of commands, global commands, which operate across all replicator services, and service-specific commands that perform operations on a specific service and/or host. For information on the global commands available, see Section 7.8.2, “trepctl Global Commands”. Information on individual commands can be found in Section 7.8.3, “trepctl Service Commands”.

### 7.8.1. trepctl 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.

```
shell> 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.

### 7.8.2. `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 7.11. `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>Shutdown</td>
<td>Shutdown the replication services cleanly</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.

#### 7.8.2.1. `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
```
7.8.2.2. `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 D.2, "Generated Field Reference".

For a replicator with multiple services, the information is output for each configured service:

```
shell> trepctl services
Processing services command...
NAME    VALUE
------   -----  
appliedLastSeqno: 44
appliedLatency  : 0.692
role        : master
serviceName : alpha
serviceType : local
started     : true
state      : ONLINE
NAME    VALUE
------   -----  
appliedLastSeqno: 40
appliedLatency  : 0.57
role        : slave
serviceName : beta
serviceType : local
started     : true
state      : ONLINE
NAME    VALUE
------   -----  
appliedLastSeqno: 41
appliedLatency  : 0.06
role        : slave
serviceName : gamma
serviceType : remote
started     : true
state      : ONLINE
Finished services command...
```

The information can be reported in JSON format by using the `-json` option to the command:

```
shell> trepctl services -json
{
  "serviceType" : "local",
  "appliedLatency" : "0.48",
  "serviceName" : "alpha",
  "appliedLastSeqno" : "2541",
  "started" : "true",
  "role" : "master",
  "state" : "ONLINE"
}
```

The information is output as an array of objects, one object for each service identified.

If the `-full` option is added, the JSON output includes full details of the service, similar to that output by the `trepctl status` command, but for each configured service:

```
shell> trepctl services -json -full
```
For more information on the fields displayed, see Section D.2, "Generated Field Reference".

7.8.2.3. trepctl shutdown Command

Deprecated in 2.0.1. This command was deprecated in 2.0.1. See Section 2.12, "Starting and Stopping Continuent Tungsten".

The shutdown command safely shuts down the replicator service, ensuring that the current transactions being applied to the database, THL writes and Continuent Tungsten specific updates to the database are correctly completed before shutting the service down.

trepctl shutdown [-y]

When executed, trepctl will ask for confirmation:

shell> trepctl shutdown
Do you really want to shutdown the replicator? [yes/NO]

The default is no. To shutdown the service without requiring interactive responses, use the -y option:

shell> trepctl shutdown -y
Replicator appears to be stopped

7.8.2.4. trepctl version Command

The trepctl version command outputs the version number of the specified replicator service.

trepctl version

shell> trepctl version
Tungsten Replicator 2.0.5 build 3

The system can also be used to obtain remote version:

shell> trepctl -host host2 version
Tungsten Replicator 2.0.5 build 3

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.
7.8.3. 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 7.12. 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>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>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>start</td>
<td>Start replication service</td>
</tr>
<tr>
<td>status</td>
<td>Print replicator status information</td>
</tr>
<tr>
<td>stop</td>
<td>Stop replication service</td>
</tr>
<tr>
<td>unload</td>
<td>Unload the replication service</td>
</tr>
<tr>
<td>unload-y</td>
<td>Unload the replication service</td>
</tr>
<tr>
<td>wait</td>
<td>Wait up to s seconds for replicator state s</td>
</tr>
</tbody>
</table>

The following sections detail each command individually, with specific options, operations and information.

7.8.3.1. trepctl backup Command

The trepctl backup command performs a backup of the corresponding database for the selected service.

```
trepctl backup [-backup agent [174]] [-limit s [174]] [-storage agent [174]]
```

Where:

Table 7.13. trepctl backup Command Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-backup agent</td>
<td>Select the backup agent</td>
</tr>
<tr>
<td>-limit s</td>
<td>The period to wait before returning after the backup request</td>
</tr>
<tr>
<td>-storage agent</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:
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 7.8.3.15, "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.

The URI can be used during a restore.

### 7.8.3.2. 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 7.8.3.3, "trepctl check Command".

• **Heartbeat**
  Indicates whether the heartbeat service is supported. For more information see Section 7.8.3.7, "trepctl heartbeat Command".

• **Flush**
  Indicates whether the `trepctl flush` operation is supported.

### 7.8.3.3. 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.

### 7.8.3.4. 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
```

### 7.8.3.5. 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>
<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> 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> trepctl clients -json

{  "rmiPort": "10000",  "rmiHost": "host4" }
```

---

Command-line Tools

- Consistency Check
  - Indicates whether the internal consistency check is supported. For more information see Section 7.8.3.3, "trepctl check Command".
- Heartbeat
  - Indicates whether the heartbeat service is supported. For more information see Section 7.8.3.7, "trepctl heartbeat Command".
- Flush
  - Indicates whether the `trepctl flush` operation is supported.

### 7.8.3.3. 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.

### 7.8.3.4. 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
```

### 7.8.3.5. 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>
<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> 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> trepctl clients -json

{  "rmiPort": "10000",  "rmiHost": "host4" }
```
The information is divided first by host, and then by the RMI management port.

### 7.8.3.6. `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> trepctl flush
```

The flush operation is always initiated, and by default `trepctl` will wait until the operation completes. Using the `-limit` option, the amount of time the command-line waits before returning can be specified:

```
shell> trepctl flush -limit 1
```

### 7.8.3.7. `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> trepctl heartbeat
```

When performing specific operations, the heartbeat can be given a name:

```
shell> trepctl heartbeat -name dataload
```

Heartbeats insert a transaction into the THL using the transaction metadata and can be used to identify whether replication is operating between replicator hosts by checking that the sequence number has been replicated to the 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> trepctl status
Processing status command...
NAME                      VALUE
----                      -----
appliedLastEventId       : mysql-bin.000009:0000000000008998;0
appliedLastSeqno         : 3630
...
```

2. Insert a heartbeat event:

```
shell> trepctl heartbeat
```

3. Check the sequence number again:

```
shell> trepctl status
Processing status command...
NAME                      VALUE
----                      -----
```
4. Check that the sequence number on the slave matches:

```
shell> trepctl status
Processing status command...
NAME                     VALUE
----                     -----  
appliedLastEventId     : mysql-bin.000009:0000000000009310;0
appliedLastSeqno       : 3631
```

Heartbeats are given implied names, but can be created with explicit names that can be tracked during specific events and operations.

For example, when loading a specific set of data, the information may be loaded and then a backup executed on the 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:

```
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:

```
NAME                     VALUE
----                     -----  
appliedLastEventId     : mysql-bin.000009:0000000000008271;0
appliedLastSeqno       : 3627
appliedLatency         : 0.704
offlineRequests        : Offline at heartbeat event: dataload
```

2. On the master:

```
master shell> mysql newdb < newdb.load
```

3. Once the data load has completed, insert the heartbeat on the master:

```
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.

### Internal Implementation

Internally, the heartbeat system operates through a tag added to the metadata of the THL entry and through a dedicated `heartbeat` table within the schema created for the replicator service. The table contains the sequence number, event ID, timestamp and heartbeat name. The heartbeat information is written into a special record within the transaction history log. A sample THL entry can be seen in the output below:

```
SEQ# = 3629 / FRAG# = 0 (last frag)
- TIME = 2013-07-19 12:14:57.0
- EPOCH# = 3614
- EVENTID = mysql-bin.000009:0000000000008681;0
- SOURCEID = host1
- METADATA = [mysql_server_id=1687011;dbms_type=mysql;is_metadata=true;service=alpha;
  shard=tungsten_alpha;heartbeat=dataload]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [##charset = UTF-8, autocommit = 1, sql_auto_is_null = 0,
  foreign_key_checks = 1, unique_checks = 1, sql_mode = 'IGNORE_SPACE',
  character_set_client = 33, collation_connection = 33, collation_server = 8]
- SCHEMA = tungsten_alpha
- SQL(0) = UPDATE tungsten_alpha.heartbeat SET source_tstamp= '2013-07-19 12:14:57',
  salt= 9, name= 'dataload' WHERE id= 1
```

During replication, 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.
7.8.3.8. `trepctl load` Command

Load the replicator service.

```
trepctl load
```

Load the replicator service. The service name must be specified on the command-line, even when only one service is configured:

```
shell> trepctl load
Operation failed: You must specify a service name using -service
```

The service name can be specified using the `-service` option:

```
shell> trepctl -service alpha load
Service loaded successfully: name=alpha
```

7.8.3.9. `trepctl offline` Command

The `trepctl offline` command puts the replicator into the offline state, stopping replication.

```
trepctl offline [-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
```

7.8.3.10. `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.

```
trepctl offline-deferred [-at-event event][-at-heartbeat [heartbeat]][-at-seqno seqno][-at-time YYYY-MM-DD_hh:mm:ss]
```

Where:

Table 7.15. `trepctl offline-deferred` Command Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-at-event event</code></td>
<td>Go offline at the specified event</td>
</tr>
<tr>
<td><code>-at-heartbeat [heartbeat]</code></td>
<td>Go offline when the specified heartbeat is identified</td>
</tr>
<tr>
<td><code>-at-seqno seqno</code></td>
<td>Go offline at the specified sequence number</td>
</tr>
<tr>
<td><code>-at-time YYYY-MM-DD_hh:mm:ss</code></td>
<td>Go offline at the specified time</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 7.8.3.9, "trepctl offline Command".

The supported triggers are:

- `-at-seqno [178]"
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** [179]

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** [179]

Specifies the name of a specific heartbeat to look for when replication should be stopped.

- **-at-time** [179]

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.

### 7.8.3.11. 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.

```
trepctl online [-base-seqno x] [force] [-from-event event] [no-checksum] [-skip-seqno x,y,z] [-until-event event] [-until-heartbeat [name]] [-until-seqno seqno] [-until-time YYYY-MM-DD_hh:mm:ss]
```

Where:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>
</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.

### 7.8.3.11.1. 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
  ```

### 7.8.3.11.2. Going Online from a Base Sequence Number

Alternatively, the base sequence number, the transaction ID where replication should start, can be specified explicitly:

```
shell> trepctl online -base-seqno 5250
```

**Warning**

Use of `-base-seqno` should be restricted to replicators in the `master` role only. Use on slaves may lead to duplication or corruption of data.

### 7.8.3.11.3. Going Online from a Specific Event

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'
```

Because events are not sequential numbers, the replicator will go online at the next nearest event id that corresponds to a transaction.
7.8.3.11.4. 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.

7.8.3.11.4.1. 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.

7.8.3.11.4.2. 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.

7.8.3.11.4.3. 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
```

7.8.3.11.4.4. 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.

7.8.3.11.5. 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
```

7.8.3.11.6. Going Online without Validating Checksum

In the event of a checksum problem in the THL, checksums can be disabled using the `-no-checksum` option:

```
shell> trepctl online -no-checksum
```

This will bring the replicator online without reading or writing checksum information.

**Important**

Use of the `-no-checksum` option disables both the reading and writing of checksums on log records. If starting the replicator without checksums to get past a checksum failure, the replicator should be taken offline again once
7.8.12. `trepctl properties` Command

Display a list of all the internal properties. The list can be filtered.

`trepctl properties [-filter name]`

The list of properties can be used to determine the current configuration:

```
shell> trepctl properties

{ "replicator.store.thl.log_file_retention": "7d",
  "replicator.filter.bidiSlave.allowBidiUnsafe": "false",
  "replicator.extractor.dbms.binlog_file_pattern": "mysql-bin",
  "replicator.filter.pkey.url": »
    "jdbc:mysql:thin://host2:3306/tungsten_alpha?createDB=true",
  ...
}
```

**Note**

Passwords are not displayed in the output.

The information is output as a JSON object with key/value pairs for each property and corresponding value.

The list can be filtered using the `--filter` option:

```
shell> trepctl properties --filter shard

{ "replicator.filter.shardfilter": »
  "com.continuent.tungsten.replicator.shard.ShardFilter",
  "replicator.filter.shardbyseqno": »
  "com.continuent.tungsten.replicator.filter.JavaScriptFilter",
  "replicator.filter.shardbyseqno.shards": "1000",
  "replicator.filter.shardfilter.enforceHome": "false",
  "replicator.filter.shardfilter.unknownShardPolicy": "error",
  "replicator.filter.shardfilter.enabled": "true",
  "replicator.filter.shardfilter.allowWhitelisted": "false",
  "replicator.shard.default.db": "stringent",
  "replicator.filter.shardbytable.script": »
    "com.continuent.tungsten.replicator.filter.JavaScriptFilter",
  "replicator.filter.shardfilter.autoCreate": "false",
  "replicator.filter.shardfilter.unwantedShardPolicy": "error"
}
```

The value or values from filtered properties can be retrieved by using the `--values` option:

```
shell> trepctl properties --filter site.name --values
default
```

If a filter that would select multiple values is specified, all the values are listed without field names:

```
shell> trepctl properties --filter shard --values
com.continuent.tungsten.replicator.shard.ShardFilter
com.continuent.tungsten.replicator.filter.JavaScriptFilter
1000
false
true
false
true
false
true
```

7.8.13. `trepctl purge` Command

Forces all logins on the attached database, other than those directly related to Continuent Tungsten, 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.
Command-line Tools

`trepctl purge [-limit s]`

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>
</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.

### 7.8.3.14. 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 [-y]`

Where:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-y</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
```

### 7.8.3.15. trepctl restore Command

Restores the database on a host from a previous backup.

`trepctl` capabilities

Once the restore has been completed, the node will remain in the **OFFLINE** state. The datasource should be switched **ONLINE** using `trepctl`.
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.

### 7.8.3.16. 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>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-role</code></td>
<td>Replicator role</td>
</tr>
<tr>
<td><code>-uri</code></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 thl://host1:2112/
```

### 7.8.3.17. 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>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-delete shard</code></td>
<td>Delete a shard definition</td>
</tr>
<tr>
<td><code>-insert shard</code></td>
<td>Add a new shard definition</td>
</tr>
<tr>
<td><code>-list</code></td>
<td>List configured shards</td>
</tr>
<tr>
<td><code>-update shard</code></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.

For more information, see Section 2.5, "Deploying a Multi-site/Multi-master Topology" Section 2.6, "Deploying Composite (SOR) Topologies".

#### 7.8.3.17.1. Listing Current Shards

To obtain a list of the currently configured shards:

```
shell> trepctl shard -list
shard_id master critical
alpha sales true
```

The shard map information can also be captured and then edited to update existing configurations:

```
shell> trepctl shard -list>shard.map
```

#### 7.8.3.17.2. 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
```
7.8.3.17.3. Updating an Existing Shard Configuration

To update a definition:

```bash
shell> trepctl shard -update < shard.map
Reading from standard input
1 shard updated
```

7.8.3.17.4. Deleting a Shard Configuration

To delete a single shard definition, specify the shard name:

```bash
shell> trepctl shard -delete alpha
```

7.8.3.18. `trepctl start` Command

Deprecated in 2.0.1. This command was deprecated in 2.0.1; use Section 7.8.3.8, "trepctl load Command".

Start the replicator service.

```
trepctl start
```

Start the replicator service. The service name must be specified on the command-line, even when only one service is configured:

```
shell> trepctl start
Operation failed: You must specify a service name using -service
```

The service name can be specified using the `-service` option:

```
shell> trepctl -service alpha start
Service started successfully: name=alpha
```

7.8.3.19. `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-assignmentsservicesshardstagestoretaskswatches ]
```

Where:

Table 7.21. `trepctl status` Command Options

<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>
</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
currentTimeMillis      : 1369233160014
dataServerHost         : host1
masterListenUri        : thl://host1:2112/  
```
For more information on the field information output, see Section D.2, "Generated Field Reference".

7.8.3.19.1. Getting Detailed Status

More detailed information about selected areas of the replicator status can be obtained by using the -name option.

7.8.3.19.1. Detailed Status: Channel Assignments

When using a single threaded replicator service, the channel-assignments will output an empty status. In parallel replication deployments, the 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
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)...
```

7.8.3.19.1.2. Detailed Status: Services

The services status output shows a list of the currently configure internal services that are defined within the replicator.

```shell
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)...
```

7.8.3.19.1.3. Detailed Status: Shards

7.8.3.19.1.4. Detailed Status: Stages

The 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
shell> trepctl status -name stages
Processing status command (stages)...
NAME     VALUE
-----     -----)
applier.class : com.continuent.tungsten.replicator.thl.THLStoreApplier
```
7.8.3.19.1. Detailed Status: Stores

The 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    : 1
extractor.class      : com.continuent.tungsten.replicator.mysql.MySQLExtractor
extractor.name       : dbms
name                 : binlog-to-q
processedMinSeqno    : 224
taskCount            : 1
NAME                  VALUE
----                  ----
applier.class        : com.continuent.tungsten.replicator.storage.InMemoryQueueAdapter
applier.name         : queue
blockCommitRowCount  : 1
committedMinSeqno    : 1
extractor.class      : com.continuent.tungsten.replicator.mysql.MySQLExtractor
extractor.name       : dbms
name                 : binlog-to-q
processedMinSeqno    : 224
taskCount            : 1
Finished status command (stages)...
```

When running parallel replication, the output shows the store name, sequence number and status information for each parallel replication channel:

```
shell> trepctl status -name stores
Processing status command (stores)...
NAME                  VALUE
----                  ----
activeSeqno          : 15
doChecksum          : false
flushIntervalMillis  : 0
```
### 7.8.3.19.1.6. Detailed Status: Tasks

The `trepcntl 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.

```
NAME VALUE
---- -----
appliedLastEventId mysql-bin.000015:0000000000001117;0
appliedLastSeqno 5271
appliedLatency 4656.176
applyTime 0.017
averageBlockSize 0.500
cancelled false
commits 10
currentBlockSize 0
currentLastEventId mysql-bin.000015:0000000000001117;0
currentLastFragno 0
currentLastSeqno 5271
eventCount 5
extractTime 0.385
filterTime 0.0
lastCommittedBlockSize 1
lastCommittedBlockTime 0.004

NAME VALUE
---- -----
appliedLastEventId mysql-bin.000015:0000000000001117;0
appliedLastSeqno 5271
appliedLatency 4656.188
applyTime 0.0
averageBlockSize 0.500
cancelled false
commits 10
currentBlockSize 0
currentLastEventId mysql-bin.000015:0000000000001117;0
currentLastFragno 0
currentLastSeqno 5271
eventCount 5
extractTime 0.406
filterTime 0.0
lastCommittedBlockSize 1
lastCommittedBlockTime 0.003
```
The list of tasks and information provided depends on the role of the host, the number of stages, and whether parallel apply is enabled.

### 7.8.3.19.1.7. Detailed Status: Watches

### 7.8.3.19.2. 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 D.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": "93",
  "transitioningTo": "",
  "offlineRequests": "NONE",
  "state": "ONLINE",
  "simpleServiceName": "alpha",
  "extensions": "",
  "pendingErrorEventId": "NONE",
  "sourceId": "host1",
  "serviceName": "alpha",
  "version": "Tungsten Replicator 2.0.5 build 3",
  "role": "master",
  "currentTimeMillis": "1369233410874",
  "masterConnectUri": "",
  "rmiPort": "10000",
  "siteName": "default",
  "pendingErrorSeqno": "-1",
  "appliedLatency": "0.53",
  "pipelineSource": "jdbc:mysql:thin://host1:3306/",
  "pendingErrorCode": "NONE",
  "maximumStoredSeqNo": "2504",
  "latestEpochNumber": "2500",
  "channels": "1",
  "appliedLastSeqno": "2504",
  "serviceType": "local",
  "seqnoType": "java.lang.Long",
  "currentEventId": "mysql-bin.000007:0000000000001353",
  "relativeLatency": "2125.873",
  "minimumStoredSeqno": "0",
  "timeInStateSeconds": "2125.372",
  "dataServerHost": "host1"
}
```
7.8.3.19.2.1. Detailed Status: Channel Assignments JSON Output

```
shell> trepctl status -name channel-assignments -json
{
  "channel": "0",
  "shard_id": "cheffy"
},
{
  "channel": "0",
  "shard_id": "tungsten_alpha"
}
```

7.8.3.19.2.2. Detailed Status: Services JSON Output

```
shell> trepctl status -name services -json
{
  "totalAssignments": "2",
  "accessFailures": "0",
  "storeClass": "com.continuent.tungsten.replicator.channel.ChannelAssignmentService",
  "name": "channel-assignment",
  "maxChannel": "0"
}
```

7.8.3.19.2.3. Detailed Status: Shards JSON Output

```
shell> trepctl status -name shards -json
{
  "stage": "q-to-dbms",
  "appliedLastEventId": "mysql-bin.000007:0000000007224342;0",
  "appliedLatency": "63.099",
  "appliedLastSeqno": "2514",
  "eventCount": "16",
  "shardId": "cheffy"
}
```

7.8.3.19.2.4. Detailed Status: Stages JSON Output

```
shell> trepctl status -name stages -json
{
  "applier.name": "thl-applier",
  "applier.class": "com.continuent.tungsten.replicator.thl.THLStoreApplier",
  "name": "remote-to-thl",
  "extractor.name": "thl-remote",
  "taskCount": "1",
  "committedMinSeqno": "2504",
  "blockCommitRowCount": "10",
  "processedMinSeqno": "-1",
  "extractor.class": "com.continuent.tungsten.replicator.thl.RemoteTHLExtractor"
},
{
  "applier.name": "parallel-q-applier",
  "applier.class": "com.continuent.tungsten.replicator.storage.InMemoryQueueAdapter",
  "name": "thl-to-q",
  "extractor.name": "thl-extractor",
  "taskCount": "1",
  "committedMinSeqno": "2504",
  "blockCommitRowCount": "10",
  "processedMinSeqno": "-1",
  "extractor.class": "com.continuent.tungsten.replicator.thl.THLStoreExtractor"
},
{
  "applier.name": "dbms",
  "applier.class": "com.continuent.tungsten.replicator.applier.MySQLDrizzleApplier",
  "filter.2.name": "bidiSlave",
  "name": "q-to-dbms",
  "extractor.name": "parallel-q-extractor",
  "filter.1.name": "pkey",
  "taskCount": "1",
  "committedMinSeqno": "2504",
  "blockCommitRowCount": "10",
  "filter.0.name": "mysqlsessions",
  "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",
  "blockCommitRowCount": "10",
  "filter.0.name": "mysqlsessions",
  "processedMinSeqno": "-1"
}
```
7.8.3.19.2.5. Detailed Status: Stores JSON Output

```shell>
trepctl status -name stores -json
{
    "logConnectionTimeout" : "28800",
    "doChecksum" : "false",
    "name" : "thl",
    "flushIntervalMillis" : "0",
    "logFileSize" : "100000000",
    "logDir" : "/opt/continuent/thl/alpha",
    "activeSeqno" : "2561",
    "readOnly" : "false",
    "timeoutMillis" : "2147483647",
    "storeClass" : "com.continuent.tungsten.replicator.thl.THL",
    "logFileRetainMillis" : "864000000",
    "maximumStoredSeqNo" : "2565",
    "minimumStoredSeqNo" : "2047",
    "fsyncOnFlush" : "false"
},
{
    "storeClass" : "com.continuent.tungsten.replicator.storage.InMemoryQueueStore",
    "maxSize" : "10",
    "storeSize" : "7",
    "name" : "parallel-queue",
    "eventCount" : "119"
}
```

7.8.3.19.2.6. Detailed Status: Tasks JSON Output

```shell>
trepctl status -name tasks -json
{
    "filterTime" : "0.0",
    "stage" : "remote-to-thl",
    "currentLastFragno" : "1",
    "taskId" : "0",
    "currentLastSeqno" : "2615",
    "state" : "extract",
    "extractTime" : "604.297",
    "applyTime" : "16.708",
    "averageBlockSize" : "0.982",
    "otherTime" : "0.017",
    "appliedLastEventId" : "mysql-bin.000007:0000000111424440;0",
    "appliedLatency" : "63.787",
    "currentLastEventId" : "mysql-bin.000007:0000000111424440;0",
    "eventCount" : "219",
    "appliedLastSeqno" : "2615",
    "cancelled" : "false"
},
{
    "filterTime" : "0.0",
    "stage" : "thl-to-q",
    "currentLastFragno" : "1",
    "taskId" : "0",
    "currentLastSeqno" : "2615",
    "state" : "extract",
    "extractTime" : "620.715",
    "applyTime" : "0.344",
    "averageBlockSize" : "1.904",
    "otherTime" : "0.006",
    "appliedLastEventId" : "mysql-bin.000007:0000000111424369;0",
    "appliedLatency" : "63.834",
    "currentLastEventId" : "mysql-bin.000007:0000000111424440;0",
    "eventCount" : "219",
    "appliedLastSeqno" : "2615",
    "cancelled" : "false"
},
{
    "filterTime" : "0.263",
    "stage" : "q-to-dbms",
    "currentLastFragno" : "1",
    "taskId" : "0",
    "currentLastSeqno" : "2614",
    "state" : "apply",
    "extractTime" : "533.471",
    "applyTime" : "61.618",
    "averageBlockSize" : "1.160"
}
```
7.8.3.19.2.7. Detailed Status: Tasks JSON Output

```json
{
    "otherTime": "24.052",
    "appliedLastEventId": "mysql-bin.000007:0000000110392640:0",
    "appliedLatency": "63.178",
    "currentLastEventId": "mysql-bin.000007:0000000110392711:0",
    "eventCount": "217",
    "appliedLastSeqno": "2614",
    "cancelled": "false"
}
```

7.8.3.20. `trepctl stop` Command

**Deprecated in 2.0.1.** This command was deprecated in 2.0.1; use Section 7.8.3.21, "`trepctl unload` Command".

Stop the replicator service.

```
trepctl stop [-y]
```

Stop the replicator service entirely. An interactive prompt is provided to confirm the shutdown:

```
shell> trepctl stop
Do you really want to stop replication service alpha? [yes/NO]
```

To disable the prompt, use the `-y` option:

```
shell> trepctl stop -y
Service stopped successfully: name=alpha
```

The name of the service stopped is provided for confirmation.

7.8.3.21. `trepctl unload` Command

Unload the replicator service.

```
trepctl unload
```

Unload the replicator service entirely. An interactive prompt is provided to confirm the shutdown:

```
shell> trepctl unload
Do you really want to unload replication service alpha? [yes/NO]
```

To disable the prompt, use the `-y` option:

```
shell> trepctl unload -y
Service unloaded successfully: name=alpha
```

The name of the service unloaded is provided for confirmation.

7.8.3.22. `trepctl wait` Command

The `trepctl wait` command waits for the replicator to enter a specific state, or for a specific sequence number to be applied to the dataserver.

```
trepctl wait [-applied seqno] [-limit s] [-state st]
```

Where:

Table 7.22: `trepctl wait` Command Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-applied seqno</code></td>
<td>Specify the sequence number to be waited for</td>
</tr>
<tr>
<td><code>-limit s</code></td>
<td>Specify the number of seconds to wait for the operation to complete</td>
</tr>
<tr>
<td><code>-state st</code></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. ONLINE), or for a specific sequence number to be applied. For example, to wait for the replicator to go into the `ONLINE` state:
Command-line Tools

```
shell> trepctl wait -state ONLINE
```

This can be useful in scripts when the state maybe changed (for example during a backup or restore operation), allowing for an operation to take place once the requested state has been reached. Once reached, `trepctl` returns with exit status 0.

To wait a specific sequence number to be applied:

```
shell> trepctl wait -applied 2000
```

This can be useful when performing bulk loads where the sequence number where the bulk load completed is known, or when waiting for a specific sequence number from the master to be applied on the slave. Unlike the offline-deferred operation, no change in the replicator is made. Instead, `trepctl` simply returns with exit status 0 when the sequence number has been successfully applied.

If the optional `-limit` [193] option is used, then `trepctl` waits for the specified number of seconds for the request event to occur. For example, to wait for 10 seconds for the replicator to go online:

```
shell> trepctl wait -state ONLINE -limit 10
```

Wait timed out!

If the requested event does not take place before the specified time limit expires, then `trepctl` returns with the message 'Wait timed out!', and an exit status of 1.

7.9. The tpm Command

tpm, or the Tungsten Package Manager, is a complete configuration, installation and deployment tool for Continuent Tungsten. 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 C, 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 Continuent Tungsten and the configuration, and optionally automatically starts the services on each host. tpm manages the entire deployment, configuration and upgrade procedure.

- **tpm INI configuration** — tpm uses an INI file to configure the service on the local host. The INI file must be create on each host that will run Continuent Tungsten. 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 7.9.1, “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 Continuent Tungsten.

- During an installation or upgrade, tpm copies the current distribution to each remote host.

- The core configuration file is translated to translate a number of template files within the configuration of each component of the system into the configuration properties files used by Continuent Tungsten. 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 Continuent Tungsten 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 Continuent Tungsten 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 Continuent Tungsten 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 7.9.5, "`tpm` Commands"; configuration options, which determine the parameters that configure individual services, which are detailed in Section 7.9.6, "`tpm` Configuration Options"; and the options that alter the way `tpm` operates, covered in Section 7.9.3, "`tpm` Staging Configuration".

### 7.9.1. Comparing Staging and `tpm` Methods

`tpm` supports two different deployment methodologies. Both configure one or more Continuent Tungsten services, in a safe and secure manner, but differ in the steps and process used to complete the installation. The two methods are:

- **Staging Directory**
  
  When using the staging directory method, a single configuration that defines all services and hosts within the Continuent Tungsten 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 7.1. tpm Staging Based Deployment](image)

- **INI File**

  When using the `ini` file method, configuration for each service must be made individually using an `ini` configuration file on each host. This is ideal for deployments where you have a configuration management system (e.g. Puppet and Chef) to manage the `ini` file. It also works very well for deployments where the configuration for each system is different from the others.
Figure 7.2. `tpm` INI Based Deployment

![Diagram of TPM deployment process]

Table 7.23. 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 Config.</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 `/opt/continuent/tungsten/tools/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/continuent-tungsten-2.0.5-3`. 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 Continuent Tungsten on your staging server.
2. On each host:
   a. Complete all the Appendix C, 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 Continuent Tungsten.

b. Complete all the Appendix C, Prerequisites.

c. Create the ini file containing your configuration.

d. Execute the tpm install command to deploy the service.

When using the staging method, upgrades and updates to the configuration must be made using tpm from the staging directory. Configuration methods can be swapped from staging to ini only by manually recreating the ini file with the new configuration and running tpm update.

7.9.2. Processing Installs and Upgrades

The tpm command is designed to coordinate the deployment activity across all hosts in a data service. 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 Continuent Tungsten and deployment files to each server

During this stage part of the Continuent Tungsten 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.

```
Validations 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 rerunning the tpm install command this check will be bypassed.

- Deploy Continuent Tungsten and write configuration files

If validation is successful, we will move on to deploying Continuent Tungsten and writing the actual configuration files. The tpm command uses a JSON file that summarizes the configuration. The Continuent Tungsten processes use many different files to store the configuration and tpm is responsible for writing them.

The /opt/continuent/releases directory will start to collect multiple directories after you have run multiple upgrades. We keep the previous versions of Continuent Tungsten 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 Continuent Tungsten configuration files by hand. This will cause future updates to fail. One of the validation checks compares the file that tpm wrote 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 Continuent Tungsten services**

  After Continuent Tungsten 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 and Tungsten Manager on all nodes
    3. Start the Tungsten Replicator and Tungsten Manager on all hosts if the services were previously running
    4. Wait for the Tungsten Manager to become responsive
    5. Stop the old Tungsten Connector and Start the new Tungsten Connector on all hosts. This step is done one host at a time so that there is always one Tungsten Connector running. If `--no-connectors` was provided on the command line then this will not occur. You must go to each server running Tungsten Connector and run `tpm promote-connector`.

### 7.9.3. 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*. 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.

```
shell> tpm configure [service_name|defaults] [tpm options] [service configuration options]
```

In addition to the Section 7.9.6, "tpm Configuration Options", the common options in Table 7.24, "tpm Common Options" may be given.

#### Table 7.24. tpm Common Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--enable-validation-check String</code></td>
<td>Remove a corresponding <code>--skip-validation-check</code> argument</td>
</tr>
<tr>
<td><code>--enable-validation-warnings String</code></td>
<td>Remove a corresponding <code>--skip-validation-warnings</code> argument</td>
</tr>
<tr>
<td><code>--property, --property=key+=value, --property=key=value, --property=key~=/match/replace/</code></td>
<td>Modify the value for key in any file that the configure script touches; <code>key=value</code> - Set key to value without evaluating template values or other rules; <code>key+=value</code> - Evaluate template values and then append value to the end of the line; <code>key~=/match/replace/</code> - Evaluate template values then execute the specified Ruby regex with sub. For example <code>--property=replicator.key~/(.*)/somevalue,\1/</code> will prepend 'somevalue' before the template value for 'replicator.key'</td>
</tr>
<tr>
<td><code>--remove-property=key</code></td>
<td>Remove a corresponding <code>--property</code> argument. Subcommands: <code>defaults</code> Modify the default values used for each data service or host Command options:</td>
</tr>
<tr>
<td><code>--skip-validation-check</code></td>
<td>Do not run the specified validation check. Validation checks are identified by the string included in the error they output.</td>
</tr>
<tr>
<td><code>--skip-validation-warnings</code></td>
<td>Do not display warnings for the specified validation check. Validation checks are identified by the string included in the warning they output.</td>
</tr>
</tbody>
</table>
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 Continuent Tungsten, the `tpm` command will only use `CONTINUENT_PROFILES`.

If you are running Tungsten Replicator, the `tpm` command will use `REPLICATOR_PROFILES` if it is available, before using `CONTINUENT_PROFILES`.

### 7.9.3.1. Configuring default options for all services

```shell
./tools/tpm configure defaults \
  --replication-user=tungsten \
  --replication-password=secret \
  --replication-port=13306
```

These options will apply to all services in the configuration file. This is useful when working with a composite dataservice or multiple independent services. These options may be overridden by calls to `tpm configure service_name` or `tpm configure service_name --hosts`.

### 7.9.3.2. Configuring a single service

```shell
./tools/tpm configure alpha \
  --master=host1 \
  --members=host1,host2,host3 \
  --home-directory=/opt/continuent \
  --user=tungsten
```

The configuration options provided following the service name will be associated with the ‘alpha’ dataservice. These options will override any given with `tpm configure defaults`.

**Relationship of `--members`, `--slaves` and `--master`**

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` and `--slaves`; `--members` will be calculated as the unique join of both values.
- If you specify `--master` and `--members`; `--slaves` will be calculated as the unique difference of both values.

### 7.9.3.3. Configuring a single host

```shell
./tools/tpm configure alpha --hosts=host3 \
  --backup-method=xtrabackup-incremental
```

This will apply the `--repl-backup-method` option to just the host3 server. Multiple hosts may be given as a comma-separated list. The names used in the `--members`, `--slaves`, `--master`, `--connectors` options should be used when calling `--hosts`. These values will override any given in `tpm configure defaults` or `tpm configure alpha`.

### 7.9.3.4. 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.

```shell
./tools/tpm reverse
```

# Defaults for all data services and hosts
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
7.9.3.5. Installation

After you have prepared the configuration file, it is time to install.

shell> ./tools/tpm install

This will install all services defined in configuration. The installation will be done as explained in Section 7.9.2, "Processing Installs and Upgrades". This will include the full set of --members [237], --slaves [248], --master [237] and --connectors [226].

7.9.3.5.1. Installing a set of specific services

shell> ./tools/tpm install alpha,bravo

All hosts included in the alpha and bravo services will be installed. The installation will be done as explained in Section 7.9.2, "Processing Installs and Upgrades".

7.9.3.5.2. Installing a set of specific hosts

shell> ./tools/tpm install --hosts=host1,host2

Only host1 and host2 will be installed. The installation will be done as explained in Section 7.9.2, "Processing Installs and Upgrades".

7.9.3.6. Upgrades from a Staging Directory

This process must be run from the staging directory in order to run properly. Determine where the current software was installed from.

shell> tpm query staging
Tungsten@staging-host:/opt/continuent/software/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 the parent directory before proceeding. Unpack the new software into the /opt/continuent/software directory and make it your current directory.

shell> tar zxf continuent-tungsten-2.0.5-3.tar.gz
shell> cd continuent-tungsten-2.0.5-3

Before any update, the current configuration must be known. If the $CONTINUENT_PROFILES [376] or $REPLICATOR_PROFILES [377] 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.

Note

The tpm update command may cause a brief outage while restarting the connectors. This will occur if you are upgrading to a new version You can avoid that with:

shell> ./tools/tpm update dataservice --no-connectors

The connectors must be updated separately on each server by running:

shell> tpm promote-connector

7.9.3.7. 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.

**Note**

The `tpm update` command may cause a brief outage while restarting the connectors. This will occur if you are upgrading to a new version. You can avoid that with:

```
shell> ./tools/tpm update dataservice --no-connectors
```

The connectors must be updated separately on each server by running:

```
shell> ./tpm promote-connector
```

### 7.9.4. 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 C, 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.

#### 7.9.4.1. 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 at `/etc/tungsten.ini` or `/etc/tungsten/tungsten.ini`. The file must be readable by the tungsten system user.

Here is an example of a `tungsten.ini` file that would setup a simple dataservice.

```ini
[defaults]
application-password=secret
application-port=3306
```
The property names in the INI file are the same as what is used on the command line. Simply remove the leading `--` characters and add it to the proper section. Each section in the INI file replaces a single `tpm configure` call. The section name inside of the square brackets is used as the service name. In the case of the `[defaults]` Section, this will act like the `tpm configure defaults` command.

Include any host-specific options in the appropriate section. This configuration will only apply to the local server, so there is no need to put host-specific settings in a different section.

### 7.9.4.2. 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.

```bash
shell> cd /opt/continuent/software/continuent-tungsten-2.0.5-3
shell> ./tools/tpm install
```

The `tpm` command will read the `tungsten.ini` file and setup all dataservices on the current server.

### 7.9.4.3. Upgrades with an INI File

Use the `tpm update` command to upgrade to the latest version.

```bash
shell> cd /opt/continuent/software
shell> tar zxf continuent-tungsten-2.0.5-3.tar.gz
shell> cd continuent-tungsten-2.0.5-3
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.

**Note**

The `tpm update` command may cause a brief outage while restarting the connectors. This will occur if you are upgrading to a new version You can avoid that with:

```bash
shell> ./tools/tpm update dataservice --no-connectors
```

The connectors must be updated separately on each server by running:

```bash
shell> tpm promote-connector
```

### 7.9.4.4. 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`.

```bash
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.

**Note**

The `tpm update` command may cause a brief outage while restarting the connectors. This will occur if you are upgrading to a new version You can avoid that with:

```bash
shell> ./tools/tpm update dataservice --no-connectors
```

The connectors must be updated separately on each server by running:

```bash
shell> tpm promote-connector
```
7.9.5. tpm Commands

All calls to `tpm` will follow a similar structure, made up of the `command`, which defines the type of operation, and one or more options.

```
shell> tpm command [sub command] [tpm options] [command options]
```

The command options will vary for each command. The core `tpm` options are:

### Table 7.25. tpm Core Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--force, -f</code></td>
<td>Do not display confirmation prompts or stop the configure process for errors</td>
</tr>
<tr>
<td><code>--help, -h</code></td>
<td>Displays help message</td>
</tr>
<tr>
<td><code>--info, -i</code></td>
<td>Display info, notice, warning and error messages</td>
</tr>
<tr>
<td><code>--log</code></td>
<td>Write all messages, visible and hidden, to this file. You may specify a</td>
</tr>
<tr>
<td></td>
<td>filename, ‘pid’ or ‘timestamp’.</td>
</tr>
<tr>
<td><code>--net-ssh-option=key=value</code></td>
<td>Set the Net::SSH option for remote system calls</td>
</tr>
<tr>
<td><code>--notice, -n</code></td>
<td>Display notice, warning and error messages</td>
</tr>
<tr>
<td><code>--preview, -p</code></td>
<td>Displays the help message and preview the effect of the command line options</td>
</tr>
<tr>
<td><code>--profile file</code></td>
<td>Sets name of config file (default: tungsten.cfg)</td>
</tr>
<tr>
<td><code>--quiet, -q</code></td>
<td>Only display warning and error messages</td>
</tr>
<tr>
<td><code>--verbose, -v</code></td>
<td>Display debug, info, notice, warning and error messages</td>
</tr>
</tbody>
</table>

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 C, Prerequisites. There are two exceptions for this:

1. When the `--hosts` 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`, `--slaves`, `--master`, `--connectors` arguments should be used when calling `--hosts`.

2. When you are using an INI configuration file (see Section 7.9.4, “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 Continuent Tungsten 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 C.1, “Staging Host Configuration” for instructions on selecting a staging directory.

### Table 7.26. 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>connector</td>
<td>Open a connection to the configured connector using mysql</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>promote-connector</td>
<td>Restart the connectors in the active configuration</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>restart</td>
<td>Restart the services on specified or added hosts</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 services on specified or added hosts</td>
</tr>
<tr>
<td>stop</td>
<td>Stop services on specified or added hosts</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>

7.9.5.1. `tpm configure` Command

The `configure` command to `tpm` creates a configuration file within the current profiles directory.

7.9.5.2. `tpm connector` Command

This will open a MySQL CLI connection to the local Tungsten Connector using the current values for `--application-user` [221], `--application-password` [220] and `--application-port` [220].

```
shell> tpm connector
```

This command will fail if the mysql utility is not available or if the local server does not have a running Tungsten Connector.

7.9.5.3. `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-diag-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`.

The structure of the created file will depend on the configured hosts, but will include all the logs for each accessible host configured. For example:

<table>
<thead>
<tr>
<th>Archive: tungsten-diag-2014-07-08-11-24-01.zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length  Date   Time   Name</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

203
There are some cases where you would like to review the configuration or make changes prior to the upgrade. In these cases it is possible to fetch the configuration and process the upgrade as different steps.

```shell
./tools/tpm fetch \
  --directory=/opt/continuent \
  --hosts=host1,autodetect
```

This will load the configuration into the local staging directory. You can then make changes using `tpm configure` before pushing out the upgrade.

The `tpm fetch` command supports the following arguments:

- **--hosts** [234]
  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** [251]
  The username to be used when logging in to other hosts.

- **--directory**
  The installation directory of the current Continuent Tungsten installation. If `autodetect` is specified, then `tpm` will look for the installation directory by checking any running Continuent Tungsten processes.

The `tpm firewall` command displays port information required to configured a firewall. When used, the information shown is for the current host:

```shell
tpm firewall
```

The information shows which ports, on which hosts, should be opened to enable communication.

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]
```

To get a list of available configuration options, use the `config-file` subcommand:

```shell
tpm help config-file
```

<table>
<thead>
<tr>
<th>Config File Options</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>config_target_basename</td>
<td>[continuent-tungsten-2.0.5-3_pid10926]</td>
</tr>
<tr>
<td>deployment_command</td>
<td>Current command being run</td>
</tr>
<tr>
<td>remote_package_path</td>
<td>Path on the server to use for running tpm commands</td>
</tr>
</tbody>
</table>
7.9.5.7. 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 Continuent Tungsten 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.

7.9.5.8. tpm mysql Command

This will open a MySQL CLI connection to the local MySQL server using the current values for --replication-user [246], --replication-password [245] and --replication-port [246].

```
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.

7.9.5.9. tpm promote-connector Command

The tpm promote-connector command should be used after performing a tpm update or tpm promote with the --no-connectors option. When using this option with these commands, running connectors are not stopped and restarted with the latest configuration or application updates, which would otherwise interrupt active applications using the connector.

The tpm promote-connector stops and restarts the configured Connector services on all configured hosts using the currently active configuration:

```
shell> ./tools/tpm promote-connector
```

NOTE >> Command successfully completed

7.9.5.10. 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
Command-line Tools

- `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 Continuent Tungsten was installed
- `tpm query topology` — return the current topology
- `tpm query usermap` — return the list of users organised by type from the `user.map`
- `tpm query values` — return the list of configured values
- `tpm query version` — get the version of the current installation

7.9.5.10.1. `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 of a JSON value:

```
shell> tpm query config
{
  "__system_defaults_will_be_overwritten__": {
  ...
  "staging_directory": "/home/tungsten/continuent-tungsten-2.0.5-3",
  "staging_host": "tr-ms1",
  "staging_user": "tungsten"
  }
}
```

7.9.5.10.2. `tpm query dataservices`

Returns the list of configured dataservices that have, or will be, installed:

```
shell> tpm query dataservices
alpha                         : PHYSICAL
```

7.9.5.10.3. `tpm query deployments`

Returns a list of all the individual deployment hosts and configuration information, returned in the form of a JSON object for each installation host:

```
shell> tpm query deployments
{
  "config_target_basename": "continuent-tungsten-2.0.5-3_pid22729",
  "dataservice_host_options": {
    "alpha": { 
      "start": "true"
    }
  ...
  "staging_directory": "/home/tungsten/continuent-tungsten-2.0.5-3",
  "staging_host": "tr-ms1",
  "staging_user": "tungsten"
  }
}
```

7.9.5.10.4. `tpm query manifest`

Returns the manifest information for the identified release of Continuent Tungsten, including the build, source and component versions, returned in the form of a JSON value:

```
shell> tpm query manifest
{
  "SVN": {
  "bristlecone": { 
    "URL": "https://bristlecone.googlecode.com/svn/trunk/bristlecone",
    "revision": 170
  },
  "connector": { 
    "URL": "svn+ssh://svn.continuent.com/svnroot/tungsten/trunk/connector",
    "revision": 9150
  },
  "fsm": { 
    "URL": "svn+ssh://svn.continuent.com/svnroot/tungsten/trunk/fsm",
    "revision": 9150
  },
  "manager": 
```


7.9.5.10.5. `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.

7.9.5.10.6. `tpm query staging`

Returns the host and directory from which the current installation was created:

```
shell> tpm query staging
/Continuent_Tungsten/2.0.5-3
```

This can be useful when the installation host and directory from which the original configuration was made need to be updated or modified.

7.9.5.10.7. `tpm query topology`

Returns the current topology and list of configured servers and roles in the form of a JSON object:

```
shell> tpm query topology
{
  "host1": "slave",
  "host2": "slave",
  "host3": "master"
}
```

7.9.5.10.8. `tpm query usermap`

Returns a summarized list of the currently configured users in the `user.map`:

```
shell> tpm query usermap
# user.map Summary
# Configured users
app_user ******** alpha
# Script entries
# DirectRead users
# Host-based routing entries
```

7.9.5.10.9. `tpm query version`

Returns the version for the identified version of Continuent Tungsten:

```
shell> tpm query version
```
7.9.5.11. \texttt{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 \texttt{--hosts} option.

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

7.9.5.12. \texttt{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 \texttt{--hosts} option.

\begin{verbatim}
shell> ./tools/tpm reset-thl
\end{verbatim}

7.9.5.13. \texttt{tpm restart} Command

The \texttt{tpm restart} command contacts the currently configured services on the current host and restarts each service. On a running system this will result in an interruption to service as the services are restarted.

The \texttt{restart} command can be useful in situations where services may not have started properly, or after a reboot services failed. For more information on explicitly starting components, see Section 2.12, "Starting and Stopping Continuent Tungsten". For information on how to configure services to start during a reboot, see Section 2.13, "Configuring Startup on Boot".

7.9.5.14. \texttt{tpm reverse} Command

The \texttt{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.

\begin{verbatim}
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=3306 \
--replication-user=tungsten \
--start-and-report=true \
--user=tungsten
# Options for the alpha data service
tools/tpm configure alpha \
--connectors=host1,host2,host3 \
--master=host1 \
--members=host1,host2,host3
\end{verbatim}

7.9.5.15. \texttt{tpm start} Command

The \texttt{tpm start} command starts configured services on the current host. This can be useful in situations where you have installed services but not configured them to be started.
The `tpm start` command can also be provided with the name of a service, which will start all the processes for that service on the current host.

See also the `tpm restart` command, Section 2.12, "Starting and Stopping Continuent Tungsten", and Section 2.13, "Configuring Startup on Boot".

7.9.16. tpm stop Command

The `tpm stop` command contacts all configured services on the current host and stops them if they are running.

See also the `tpm restart` command, Section 2.12, "Starting and Stopping Continuent Tungsten", and Section 2.13, "Configuring Startup on Boot".

7.9.17. 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 7.9.2, "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 `/opt/continuent/tungsten/tools/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/continuent-tungsten-2.0.5-3`. 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 7.9.3.6, "Upgrades from a Staging Directory".

If an INI file was used; see Section 7.9.4.3, "Upgrades with an INI File"

Applying configuration changes to the current version

If a staging directory was used; see Section 7.9.3.7, "Configuration Changes from a Staging Directory".

If an INI file was used; see Section 7.9.4.4, "Configuration Changes with an INI file".

7.9.5.18. `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
#
#
...
```

The command can be run after performing a `tpm configure` and before a `tpm install` to ensure that any prerequisite or configuration issues are addressed before installation occurs.

7.9.5.19. `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`.

7.9.6. `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` [251]

- In an INI file, without the double-dash prefix, i.e. `repl-thl-log-retention=3d` [251]
A full list of all the available options supported is provided in Table 7.27, "tpm Configuration Options".

Table 7.27. tpm Configuration Options

<table>
<thead>
<tr>
<th>Command-line Option</th>
<th>INI File Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--allow-bidi-unsafe</td>
<td>allow-bidi-unsafe</td>
<td>Allow unsafe SQL from remote service</td>
</tr>
<tr>
<td>--api</td>
<td>api</td>
<td>Enable the replication API</td>
</tr>
<tr>
<td>--api-host</td>
<td>api-host</td>
<td>Hostname that the replication API should listen on</td>
</tr>
<tr>
<td>--api-password</td>
<td>api-password</td>
<td>HTTP basic auth password for the replication API</td>
</tr>
<tr>
<td>--api-port</td>
<td>api-port</td>
<td>Port that the replication API should bind to</td>
</tr>
<tr>
<td>--api-user</td>
<td>api-user</td>
<td>HTTP basic auth username for the replication API</td>
</tr>
<tr>
<td>--application-password</td>
<td>application-password</td>
<td>Database password for the connector</td>
</tr>
<tr>
<td>--application-port</td>
<td>application-port</td>
<td>Port for the connector to listen on</td>
</tr>
<tr>
<td>--application-readonly-port</td>
<td>application-readonly-port</td>
<td>Port for the connector to listen for read-only connections on</td>
</tr>
<tr>
<td>--application-user</td>
<td>application-user</td>
<td>Database username for the connector</td>
</tr>
<tr>
<td>--auto-enable</td>
<td>auto-enable</td>
<td>Auto-enable services after start-up</td>
</tr>
<tr>
<td>--auto-recovery-delay-interval</td>
<td>auto-recovery-delay-interval</td>
<td>Delay between going OFFLINE and attempting to go ONLINE</td>
</tr>
<tr>
<td>--auto-recovery-max-attempts</td>
<td>auto-recovery-max-attempts</td>
<td>Maximum number of attempts at automatic recovery</td>
</tr>
<tr>
<td>--auto-recovery-reset-interval</td>
<td>auto-recovery-reset-interval</td>
<td>Delay before autorecovery is deemed to have succeeded</td>
</tr>
<tr>
<td>--backup-directory</td>
<td>backup-directory</td>
<td>Permanent backup storage directory</td>
</tr>
<tr>
<td>--backup-dump-directory</td>
<td>backup-dump-directory</td>
<td>Backup temporary dump directory</td>
</tr>
<tr>
<td>--backup-method</td>
<td>backup-method</td>
<td>Database backup method</td>
</tr>
<tr>
<td>--backup-online</td>
<td>backup-online</td>
<td>Does the backup script support backing up a datasource while it is ONLINE</td>
</tr>
<tr>
<td>--backup-retention</td>
<td>backup-retention</td>
<td>Number of backups to retain</td>
</tr>
<tr>
<td>--backup-script</td>
<td>backup-script</td>
<td>What is the path to the backup script</td>
</tr>
<tr>
<td>--batch-enabled</td>
<td>batch-enabled</td>
<td>Should the replicator service use a batch applier</td>
</tr>
<tr>
<td>--batch-load-language</td>
<td>batch-load-language</td>
<td>Which script language to use for batch loading</td>
</tr>
<tr>
<td>--batch-load-template</td>
<td>batch-load-template</td>
<td>Value for the loadBatchTemplate property</td>
</tr>
<tr>
<td>--channels</td>
<td>channels</td>
<td>Number of replication channels to use for services</td>
</tr>
<tr>
<td>--composite-datasources</td>
<td>composite-datasources</td>
<td>Data services that should be added to this composite data service</td>
</tr>
<tr>
<td>--datasources</td>
<td>datasources</td>
<td>Data services that should be added to this composite data service</td>
</tr>
<tr>
<td>--dataservice-composite-datasources</td>
<td>dataservice-composite-datasources</td>
<td>Data services that should be added to this composite data service</td>
</tr>
<tr>
<td>--config-file-help</td>
<td>config-file-help</td>
<td>Display help information for content of the config 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>--conn-java-mem-size [224]</td>
<td>conn-java-mem-size [224]</td>
<td>Connector Java heap memory size used to buffer data between clients and databases</td>
</tr>
<tr>
<td>--conn-round-robin-include-master [224]</td>
<td>conn-round-robin-include-master [224]</td>
<td>Should the Connector include the master in round-robin load balancing</td>
</tr>
<tr>
<td>--connector-autoreconnect [224]</td>
<td>connector-autoreconnect [224]</td>
<td>Enable auto-reconnect in the connector</td>
</tr>
<tr>
<td>--connector-delete-user-map [224]</td>
<td>connector-delete-user-map [224]</td>
<td>Overwrite an existing user.map file</td>
</tr>
<tr>
<td>--connector-drop-after-max-connections [224]</td>
<td>connector-drop-after-max-connections [224]</td>
<td>Instantly drop connections that arrive after --connector-max-connections has been reached</td>
</tr>
<tr>
<td>--connector-listen-interface [224]</td>
<td>connector-listen-interface [224]</td>
<td>Listen interface to use for the connector</td>
</tr>
<tr>
<td>--connector-max-connections [225]</td>
<td>connector-max-connections [225]</td>
<td>The maximum number of connections the connector should allow at any time</td>
</tr>
<tr>
<td>--connector-max-slave-latency [225], --connector-max-apply-latency [225]</td>
<td>connector-max-slave-latency [225], connector-max-apply-latency [225]</td>
<td>The maximum applied latency for slave connections</td>
</tr>
<tr>
<td>--connector-readonly [225], --enable-connector-readonly [225]</td>
<td>connector-readonly [225], enable-connector-readonly [225]</td>
<td>Enable the Tungsten Connector read-only mode</td>
</tr>
<tr>
<td>--connector-ro-addresses [225]</td>
<td>connector-ro-addresses [225]</td>
<td>Connector addresses that should receive a r/o connection</td>
</tr>
<tr>
<td>--connector-rw-addresses [225]</td>
<td>connector-rw-addresses [225]</td>
<td>Connector addresses that should receive a r/w connection</td>
</tr>
<tr>
<td>--connector-smartscale-sessionid [225]</td>
<td>connector-smartscale-sessionid [225]</td>
<td>Enable SmartScale R/W splitting in the connector</td>
</tr>
<tr>
<td>--connectors [226], --dataservice-connectors [226]</td>
<td>connectors [226], dataservice-connectors [226]</td>
<td>Hostnames for the dataservice connectors</td>
</tr>
<tr>
<td>--consistency-policy [226], --repl-consistency-policy [226]</td>
<td>consistency-policy [226], repl-consistency-policy [226]</td>
<td>Should the replicator stop or warn if a consistency check fails?</td>
</tr>
<tr>
<td>--datasource-name [226]</td>
<td>datasource-name [226]</td>
<td>Limit the command to the hosts in this datasource Multiple data services may be specified by providing a comma separated list</td>
</tr>
<tr>
<td>--dataservice-relay-enabled [226]</td>
<td>dataservice-relay-enabled [226]</td>
<td>Make this datasource the slave of another</td>
</tr>
<tr>
<td>--dataservice-schema [226]</td>
<td>dataservice-schema [226]</td>
<td>The db schema to hold datasource details</td>
</tr>
<tr>
<td>--dataservice-thl-port [226]</td>
<td>datasource-thl-port [226]</td>
<td>Port to use for THL operations</td>
</tr>
<tr>
<td>--dataservice-vip-enabled [226]</td>
<td>datasource-vip-enabled [226]</td>
<td>Is VIP management enabled?</td>
</tr>
<tr>
<td>--dataservice-vip-ipaddress [227]</td>
<td>datasource-vip-ipaddress [227]</td>
<td>VIP IP address</td>
</tr>
<tr>
<td>--datasource-boot-script [227], --repl-datasource-boot-script [227]</td>
<td>datasource-boot-script [227], repl-datasource-boot-script [227]</td>
<td>Database start script</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>--datasource-mysql-ibdata-directory [228], --repl-datasource-mysql-ibdata-directory [228]</td>
<td>datasource-mysql-ibdata-directory [228], repl-datasource-mysql-ibdata-directory [228]</td>
<td>MySQL InnoDB data directory</td>
</tr>
<tr>
<td>--datasource-mysql-iblog-directory [228], --repl-datasource-mysql-iblog-directory [228]</td>
<td>datasource-mysql-iblog-directory [228], repl-datasource-mysql-iblog-directory [228]</td>
<td>MySQL InnoDB log directory</td>
</tr>
<tr>
<td>--datasource-oracle-scan [228], --repl-datasource-oracle-scan [228]</td>
<td>datasource-oracle-scan [228], repl-datasource-oracle-scan [228]</td>
<td>Oracle SCAN</td>
</tr>
<tr>
<td>--datasource-oracle-service [228], --repl-datasource-oracle-service [228]</td>
<td>datasource-oracle-service [228], repl-datasource-oracle-service [228]</td>
<td>Oracle Service</td>
</tr>
<tr>
<td>--datasource-pg-archive [228], --repl-datasource-pg-archive [228]</td>
<td>datasource-pg-archive [228], repl-datasource-pg-archive [228]</td>
<td>PostgreSQL archive location</td>
</tr>
<tr>
<td>--datasource-pg-conf [228], --repl-datasource-pg-conf [228]</td>
<td>datasource-pg-conf [228], repl-datasource-pg-conf [228]</td>
<td>Location of postgresql.conf</td>
</tr>
<tr>
<td>--datasource-pg-home [228], --repl-datasource-pg-home [228]</td>
<td>datasource-pg-home [228], repl-datasource-pg-home [228]</td>
<td>PostgreSQL data directory</td>
</tr>
<tr>
<td>--datasource-type [229], --repl-datasource-type [229]</td>
<td>datasource-type [229], repl-datasource-type [229]</td>
<td>Database type</td>
</tr>
<tr>
<td>--delete [229]</td>
<td>delete [229]</td>
<td>Delete the named data service from the configuration Data Service options:</td>
</tr>
<tr>
<td>--deploy-current-package [229]</td>
<td>deploy-current-package [229]</td>
<td>Deploy the current Tungsten package</td>
</tr>
<tr>
<td>--deploy-package-uri [229]</td>
<td>deploy-package-uri [229]</td>
<td>URL for the Tungsten package to deploy</td>
</tr>
<tr>
<td>--direct-datasource-host [229], --repl-direct-datasource-host [229]</td>
<td>direct-datasource-host [229], repl-direct-datasource-host [229]</td>
<td>Database server hostname</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>--direct-datasource-type</code> [230], <code>--repl-direct-datasource-type</code> [230]</td>
<td><code>direct-datasource-type</code> [230], <code>repl-direct-datasource-type</code> [230]</td>
<td>Database type</td>
</tr>
<tr>
<td><code>--direct-replication-password</code> [231], <code>--direct-datasource-password</code> [231], <code>--repl-direct-datasource-password</code> [231]</td>
<td><code>direct-datasource-password</code> [231], <code>direct-replication-password</code> [231], <code>repl-direct-datasource-password</code> [231]</td>
<td>Database password</td>
</tr>
<tr>
<td><code>--direct-replication-user</code> [231], <code>--direct-datasource-user</code> [231], <code>--repl-direct-datasource-user</code> [231]</td>
<td><code>direct-datasource-user</code> [231], <code>direct-replication-user</code> [231], <code>repl-direct-datasource-user</code> [231]</td>
<td>Database login for Tungsten</td>
</tr>
<tr>
<td><code>--enable-active-witnesses</code> [231], <code>--active-witnesses</code> [231]</td>
<td><code>active-witnesses</code> [231], <code>enable-active-witnesses</code> [231]</td>
<td>Enable active witness hosts</td>
</tr>
<tr>
<td><code>--enable-connector-bridge-mode</code> [231], <code>--connector-bridge-mode</code> [231]</td>
<td><code>connector-bridge-mode</code> [231], <code>enable-connector-bridge-mode</code> [231]</td>
<td>Enable the Tungsten Connector bridge mode</td>
</tr>
<tr>
<td><code>--enable-connector-ssl</code> [231], <code>--connector-ssl</code> [231]</td>
<td><code>connector-ssl</code> [231], <code>enable-connector-ssl</code> [231]</td>
<td>Enable SSL encryption of connector traffic to the database</td>
</tr>
<tr>
<td><code>--enable-rmi-authentication</code> [232], <code>--rmi-authentication</code> [232]</td>
<td><code>enable-rmi-authentication</code> [232], <code>rmi-authentication</code> [232]</td>
<td>Enable RMI authentication for the services running on this host</td>
</tr>
<tr>
<td><code>--enable-rmi-ssl</code> [232], <code>--rmi-ssl</code> [232]</td>
<td><code>enable-rmi-ssl</code> [232], <code>rmi-ssl</code> [232]</td>
<td>Enable SSL encryption of RMI communication on this host</td>
</tr>
<tr>
<td><code>--enable-sudo-access</code> [233], <code>--root-command-prefix</code> [233]</td>
<td><code>enable-sudo-access</code> [233], <code>root-command-prefix</code> [233]</td>
<td>Run root commands using sudo</td>
</tr>
<tr>
<td><code>--enable-thl-ssl</code> [233], <code>--repl-enable-thl-ssl</code> [233], <code>--thl-ssl</code> [233]</td>
<td><code>enable-thl-ssl</code> [233], <code>repl-enable-thl-ssl</code> [233], <code>thl-ssl</code> [233]</td>
<td>Enable SSL encryption of THL communication for this service</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>--force [233]</code></td>
<td><code>r [233], force [233]</code></td>
<td>Do not display confirmation prompts or stop the configure process for errors</td>
</tr>
<tr>
<td><code>--help [233]</code></td>
<td><code>h [233], help [233]</code></td>
<td>Displays help message</td>
</tr>
<tr>
<td><code>--host-name [233]</code></td>
<td><code>host-name [233]</code></td>
<td>DNS hostname</td>
</tr>
<tr>
<td><code>--hosts [234]</code></td>
<td></td>
<td>Limit the command to the hosts listed You must use the hostname as it appears in the configuration.</td>
</tr>
<tr>
<td><code>--hub [234], --dataservice-hub-host [234]</code></td>
<td><code>dataservice-hub-host [234], hub [234]</code></td>
<td>What is the hub host for this all-masters dataservice?</td>
</tr>
<tr>
<td><code>--hub-service [234], --dataservice-hub-service [234]</code></td>
<td><code>dataservice-hub-service [234], hub-service [234]</code></td>
<td>The data service to use for the hub of a star topology</td>
</tr>
<tr>
<td><code>--info [234], -i [234]</code></td>
<td><code>i [234], info [234]</code></td>
<td>Display info, notice, warning and error messages</td>
</tr>
<tr>
<td><code>--install [234]</code></td>
<td><code>install [234]</code></td>
<td>Install service start scripts</td>
</tr>
<tr>
<td><code>--install-directory [234], --home-directory [234]</code></td>
<td><code>home-directory [234], install-directory [234]</code></td>
<td>Installation directory</td>
</tr>
<tr>
<td><code>--java-file-encoding [235], --repl-java-file-encoding [235]</code></td>
<td><code>java-file-encoding [235], repl-java-file-encoding [235]</code></td>
<td>Java platform charset (esp. for heterogeneous replication)</td>
</tr>
<tr>
<td><code>--java-keystore-path [236]</code></td>
<td><code>java-keystore-path [236]</code></td>
<td>Local path to the Java Keystore file.</td>
</tr>
<tr>
<td><code>--java-mem-size [236], --repl-java-mem-size [236]</code></td>
<td><code>java-mem-size [236], repl-java-mem-size [236]</code></td>
<td>Replicator Java heap memory size in Mb (min 128)</td>
</tr>
<tr>
<td><code>--java-passwordstore-path [236]</code></td>
<td><code>java-passwordstore-path [236]</code></td>
<td>Local path to the Java Password Store File.</td>
</tr>
<tr>
<td><code>--java-truststore-password [236]</code></td>
<td><code>java-truststore-password [236]</code></td>
<td>The password for unlocking the tungsten_truststore.jks file in the security directory</td>
</tr>
<tr>
<td><code>--java-truststore-path [236]</code></td>
<td><code>java-truststore-path [236]</code></td>
<td>Local path to the Java Truststore file.</td>
</tr>
<tr>
<td><code>--java-user-timezone [236], --repl-java-user-timezone [236]</code></td>
<td><code>java-user-timezone [236], repl-java-user-timezone [236]</code></td>
<td>Java VM Timezone (esp. for cross-site replication)</td>
</tr>
<tr>
<td><code>--log [236]</code></td>
<td><code>log [236]</code></td>
<td>Write all messages, visible and hidden, to this file. You may specify a filename, ‘pid’ or ‘timestamp’.</td>
</tr>
<tr>
<td><code>--log-slave-updates [236]</code></td>
<td><code>log-slave-updates [236]</code></td>
<td>Should slaves log updates to binlog</td>
</tr>
<tr>
<td><code>--master [237], --dataservice-master-host [237], --masters [237]</code></td>
<td><code>dataservice-master-host [237], master [237], masters [237]</code></td>
<td>What is the master host for this dataservice?</td>
</tr>
<tr>
<td><code>--master-preferred-role [237], --repl-master-preferred-role [237]</code></td>
<td><code>master-preferred-role [237], repl-master-preferred-role [237]</code></td>
<td>Preferred role for master THL when connecting as a slave (master, slave, etc.)</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>--master-services</code> [237], <code>--datadast-service-master-services</code> [237]</td>
<td><code>datadast-service-master-services</code> [237], <code>master-services</code> [237]</td>
<td>Data service names that should be used on each master</td>
</tr>
<tr>
<td><code>--members</code> [237], <code>--datadast-service-hosts</code> [237]</td>
<td><code>datadast-service-hosts</code> [237], <code>members</code> [237]</td>
<td>Hostnames for the datadast-service members</td>
</tr>
<tr>
<td><code>--mgr-api</code> [237]</td>
<td><code>mgr-api</code> [237]</td>
<td>Enable the Manager API</td>
</tr>
<tr>
<td><code>--mgr-api-address</code> [237]</td>
<td><code>mgr-api-address</code> [237]</td>
<td>Address for the Manager API</td>
</tr>
<tr>
<td><code>--mgr-api-port</code> [237]</td>
<td><code>mgr-api-port</code> [237]</td>
<td>Port for the Manager API</td>
</tr>
<tr>
<td><code>--mgr-group-communication-port</code> [238]</td>
<td><code>mgr-group-communication-port</code> [238]</td>
<td>Port to use for manager group communication</td>
</tr>
<tr>
<td><code>--mgr-listen-interface</code> [238]</td>
<td><code>mgr-listen-interface</code> [238]</td>
<td>Listen interface to use for the manager</td>
</tr>
<tr>
<td><code>--mgr-policy-mode</code> [238]</td>
<td><code>mgr-policy-mode</code> [238]</td>
<td>Manager policy mode</td>
</tr>
<tr>
<td><code>--mgr-remote-port</code> [238]</td>
<td><code>mgr-remote-port</code> [238]</td>
<td>Port to use for the manager RMI server</td>
</tr>
<tr>
<td><code>--mgr-ro-slave</code> [239]</td>
<td><code>mgr-ro-slave</code> [239]</td>
<td>Make slaves read-only</td>
</tr>
<tr>
<td><code>--mgr-arp-path</code> [239]</td>
<td><code>mgr-arp-path</code> [239]</td>
<td>Path to the arp binary</td>
</tr>
<tr>
<td><code>--mgr-vip-ifconfig-path</code> [239]</td>
<td><code>mgr-vip-ifconfig-path</code> [239]</td>
<td>Path to the ifconfig binary</td>
</tr>
<tr>
<td><code>--mgr-wait-for-members</code> [239]</td>
<td><code>mgr-wait-for-members</code> [239]</td>
<td>Wait for all datasources to be available before completing installation</td>
</tr>
<tr>
<td><code>--mysql-connectorj-path</code> [239]</td>
<td><code>mysql-connectorj-path</code> [239]</td>
<td>Path to MySQL Connector/J</td>
</tr>
<tr>
<td><code>--mysql-driver</code> [239]</td>
<td><code>mysql-driver</code> [239]</td>
<td>MySQL Driver Vendor</td>
</tr>
<tr>
<td><code>--mysql-enable-ansiquotes</code> [239], <code>repl-mysql-enable-ansiquotes</code> [239]</td>
<td><code>mysql-enable-ansiquotes</code> [239], <code>repl-mysql-enable-ansiquotes</code> [239]</td>
<td>Enables ANSI_QUOTES mode for incoming events?</td>
</tr>
<tr>
<td><code>--mysql-enable-settostring</code> [240], <code>repl-mysql-enable-settostring</code> [240]</td>
<td><code>mysql-enable-settostring</code> [240], <code>repl-mysql-enable-settostring</code> [240]</td>
<td>Decode SET values into their text values?</td>
</tr>
<tr>
<td><code>--mysql-ro-slave</code> [240], <code>repl-mysql-ro-slave</code> [240]</td>
<td><code>mysql-ro-slave</code> [240], <code>repl-mysql-ro-slave</code> [240]</td>
<td>Slaves are read-only</td>
</tr>
<tr>
<td><code>--mysql-server-id</code> [240], <code>repl-mysql-server-id</code> [240]</td>
<td><code>mysql-server-id</code> [240], <code>repl-mysql-server-id</code> [240]</td>
<td>Explicitly set the MySQL server ID</td>
</tr>
<tr>
<td><code>--mysql-use-bytes-for-string</code> [240], <code>repl-mysql-use-bytes-for-string</code> [240]</td>
<td><code>mysql-use-bytes-for-string</code> [240], <code>repl-mysql-use-bytes-for-string</code> [240]</td>
<td>Transfer strings as their byte representation?</td>
</tr>
<tr>
<td><code>--net-ssh-option=key=value</code> [241]</td>
<td><code>net-ssh-option=key=value</code> [241]</td>
<td>Set the Net::SSH option for remote system calls</td>
</tr>
<tr>
<td><code>--no-deployment</code> [241]</td>
<td><code>no-deployment</code> [241]</td>
<td>Skip deployment steps that create the install directory</td>
</tr>
</tbody>
</table>
### Command-line Tools

<table>
<thead>
<tr>
<th>CmdLine Option</th>
<th>INI File Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--no-validation [241]</td>
<td>no-validation [241]</td>
<td>Skip validation checks that run on each host</td>
</tr>
<tr>
<td>--notice [241], -n [241]</td>
<td>n [241], notice [241]</td>
<td>Display notice, warning and error messages</td>
</tr>
<tr>
<td>--pg-ctl [241], --repl-pg-ctl [241]</td>
<td>pg-ctl [241], repl-pg-ctl [241]</td>
<td>Path to the pg_ctl script</td>
</tr>
<tr>
<td>--pg-standby [242], --repl-pg-standby [242]</td>
<td>pg-standby [242], repl-pg-standby [242]</td>
<td>Path to the pg_standby script</td>
</tr>
<tr>
<td>--postgresql-dname [242], --repl-postgresql-dname [242]</td>
<td>postgresql-dname [242], repl-postgresql-dname [242]</td>
<td>Name of the database to replicate</td>
</tr>
<tr>
<td>--postgresql-slonik [242], --repl-postgresql-slonik [242]</td>
<td>postgresql-slonik [242], repl-postgresql-slonik [242]</td>
<td>Path to the slonik executable</td>
</tr>
<tr>
<td>--preferred-path [242]</td>
<td>preferred-path [242]</td>
<td>Additional command path</td>
</tr>
<tr>
<td>--prefetch-enabled [243]</td>
<td>prefetch-enabled [243]</td>
<td>Should the replicator service be setup as a prefetch applier</td>
</tr>
<tr>
<td>--prefetch-max-time-ahead [243]</td>
<td>prefetch-max-time-ahead [243]</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 [243]</td>
<td>prefetch-min-time-ahead [243]</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 [243]</td>
<td>prefetch-sleep-time [243]</td>
<td>How long to wait when the prefetch applier gets too far ahead</td>
</tr>
<tr>
<td>--preview [243], -p [243]</td>
<td>p [243], preview [243]</td>
<td>Displays the help message and preview the effect of the command line options</td>
</tr>
<tr>
<td>--profile file [244]</td>
<td>profile file [244]</td>
<td>Sets names of config file (default: tungsten.cfg)</td>
</tr>
<tr>
<td>--profile-script [244]</td>
<td>profile script [244]</td>
<td>Append commands to include env.sh in this profile script</td>
</tr>
<tr>
<td>--property [244], --property=key+=value [244], --property=key=value [244], --property=key=/match/replace/ [244]</td>
<td>property [244], property=key+=value [244], property=key=value [244], property=key=/match/replace/ [244]</td>
<td>Modify the value for key in any file that the configure script touches; key=value - Set key to value without evaluating template values or other rules; key+=value - Evaluate template values and then append value to the end of the line; key~=/match/replace/ - Evaluate template values then execute the specified Ruby regex with sub. For example --property=replicator.key=~/(.*)/ somevalue\1/ will prepend 'somevalue' before the template value for 'replicator.key'</td>
</tr>
<tr>
<td>--protect-configuration-files [244]</td>
<td>protect-configuration-files [244]</td>
<td>When enabled, configuration files are protected to be only readable and updateable by the configured user</td>
</tr>
<tr>
<td>--quiet [245], -q [245]</td>
<td>q [245], quiet [245]</td>
<td>Only display warning and error messages</td>
</tr>
<tr>
<td>--relay-directory [245], --repl-relay-directory [245]</td>
<td>relay-directory [245], repl-relay-directory [245]</td>
<td>Directory for logs transferred from the master</td>
</tr>
<tr>
<td>--relay-enabled [245]</td>
<td>relay-enabled [245]</td>
<td>Should the replicator service be setup as a relay master</td>
</tr>
<tr>
<td>--relay-source [245], --dataservice-relay-source [245]</td>
<td>dataservice-relay-source [245], relay-source [245]</td>
<td>Datataservice name to use as a relay source</td>
</tr>
</tbody>
</table>
| --remove-property=key [245] | remove-property=key [245] | Remove a corresponding --property argument. Subcommands: defaults Modify the default values used for each data service or host Command options:
<table>
<thead>
<tr>
<th>CmdLine Option</th>
<th>INI File Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--replication-password, --datasource-password, --repl-datasource-password</td>
<td>datasource-password, repl-datasource-password, replication-password</td>
<td>Database password</td>
</tr>
<tr>
<td>--replication-port, --datasource-port, --repl-datasource-port</td>
<td>datasource-port, repl-datasource-port, replication-port</td>
<td>Database server port</td>
</tr>
<tr>
<td>--replication-user, --datasource-user, --repl-datasource-user</td>
<td>datasource-user, repl-datasource-user, replication-user</td>
<td>Database login for Tungsten</td>
</tr>
<tr>
<td>--reset</td>
<td>reset</td>
<td>Clear the current configuration before processing any arguments</td>
</tr>
<tr>
<td>--rmi-port, --repl-rmi-port</td>
<td>repl-rmi-port, rmi-port</td>
<td>Replication RMI listen port</td>
</tr>
<tr>
<td>--rmi-user</td>
<td>rmi-user</td>
<td>The username for RMI authentication</td>
</tr>
<tr>
<td>--role, --repl-role</td>
<td>repl-role, role</td>
<td>What is the replication role for this service?</td>
</tr>
<tr>
<td>--router-gateway-port</td>
<td>router-gateway-port</td>
<td>The router gateway port</td>
</tr>
<tr>
<td>--router-jmx-port</td>
<td>router-jmx-port</td>
<td>The router jmx port</td>
</tr>
<tr>
<td>--security-directory</td>
<td>security-directory</td>
<td>Storage directory for the Java security/encryption files</td>
</tr>
<tr>
<td>--service-alias, --dataservice-alias</td>
<td>dataservice-alias</td>
<td>Replication alias of this dataservice</td>
</tr>
<tr>
<td>--service-type, --repl-service-type</td>
<td>repl-service-type, service-type</td>
<td>What is the replication service type?</td>
</tr>
<tr>
<td>--skip-statemap</td>
<td>skip-statemap</td>
<td>Do not copy the cluster-home/conf/statemap.properties from the previous install</td>
</tr>
<tr>
<td>--skip-validation-check</td>
<td>skip-validation-check</td>
<td>Do not run the specified validation check. Validation checks are identified by the string included in the error they output.</td>
</tr>
<tr>
<td>--skip-validation-warnings</td>
<td>skip-validation-warnings</td>
<td>Do not display warnings for the specified validation check. Validation checks are identified by the string included in the warning they output.</td>
</tr>
<tr>
<td>--slave-privileged-updates</td>
<td>slave-privileged-updates</td>
<td>Does login for slave update have superuser privileges</td>
</tr>
<tr>
<td>--slaves, --dataservice-slaves</td>
<td>dataservice-slaves, slaves</td>
<td>What are the slaves for this dataservice?</td>
</tr>
<tr>
<td>--start</td>
<td>start</td>
<td>Start the services after configuration</td>
</tr>
<tr>
<td>--start-and-report</td>
<td>start-and-report</td>
<td>Start the services and report out the status after configuration</td>
</tr>
<tr>
<td>--svc-allow-any-remote-service, --repl-svc-allow-any-remote-service</td>
<td>repl-svc-allow-any-remote-service, svc-allow-any-remote-service</td>
<td>Replicate from any service</td>
</tr>
<tr>
<td>--svc-applier-buffer-size, --repl-svc-applier-buffer-size</td>
<td>repl-buffer-size, repl-svc-applier-buffer-size, svc-applier-buffer-size</td>
<td>Applier block commit size (min 1)</td>
</tr>
<tr>
<td>--svc-applier-filters, --repl-svc-applier-filters</td>
<td>repl-svc-applier-filters, svc-applier-filters</td>
<td>Replication service applier filters</td>
</tr>
<tr>
<td>--svc-extractor-filters, --repl-svc-extractor-filters</td>
<td>repl-svc-extractor-filters, svc-extractor-filters</td>
<td>Replication service extractor filters</td>
</tr>
<tr>
<td>--svc-parallelization-type, --repl-svc-parallelization-type</td>
<td>repl-svc-parallelization-type, svc-parallelization-type</td>
<td>Method for implementing parallel apply</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>--svc-shard-default-db</td>
<td>repl-svc-shard-default-db</td>
<td>Mode for setting the shard ID from the default db</td>
</tr>
<tr>
<td>--repl-svc-shard-default-db</td>
<td>repl-svc-shard-default-db, svc-shard-default-db</td>
<td></td>
</tr>
<tr>
<td>--svc-table-engine</td>
<td>repl-svc-table-engine</td>
<td>Replication service table engine</td>
</tr>
<tr>
<td>--repl-svc-table-engine</td>
<td>repl-svc-table-engine, svc-table-engine</td>
<td></td>
</tr>
<tr>
<td>--svc-thl-filters</td>
<td>repl-svc-thl-filters</td>
<td>Replication service THL filters</td>
</tr>
<tr>
<td>--repl-svc-thl-filters</td>
<td>repl-svc-thl-filters, svc-thl-filters</td>
<td></td>
</tr>
<tr>
<td>--target-dataservice</td>
<td>slave-dataservice</td>
<td>Dataservice to use to determine the value of host configuration</td>
</tr>
<tr>
<td>--slave-dataservice</td>
<td>target-dataservice</td>
<td></td>
</tr>
<tr>
<td>--temp-directory</td>
<td>temp-directory</td>
<td>Temporary Directory</td>
</tr>
<tr>
<td>--template-file-help</td>
<td>template-file-help</td>
<td>Display the keys that may be used in configuration template files</td>
</tr>
<tr>
<td>--thl-directory</td>
<td>repl-thl-directory</td>
<td>Replicator log directory</td>
</tr>
<tr>
<td>--repl-thl-directory</td>
<td>repl-thl-directory, thl-directory</td>
<td></td>
</tr>
<tr>
<td>--thl-do-checksum</td>
<td>repl-thl-do-checksum</td>
<td>Execute checksum operations on THL log files</td>
</tr>
<tr>
<td>--repl-thl-do-checksum</td>
<td>repl-thl-do-checksum, thl-do-checksum</td>
<td></td>
</tr>
<tr>
<td>--thl-interface</td>
<td>repl-thl-interface</td>
<td>Listen interface to use for THL operations</td>
</tr>
<tr>
<td>--repl-thl-interface</td>
<td>repl-thl-interface, thl-interface</td>
<td></td>
</tr>
<tr>
<td>--thl-log-connection-timeout</td>
<td>repl-thl-log-connection-timeout</td>
<td>Number of seconds to wait for a connection to the THL log</td>
</tr>
<tr>
<td>--repl-thl-log-connection-timeout</td>
<td>repl-thl-log-connection-timeout, thl-log-connection-timeout</td>
<td></td>
</tr>
<tr>
<td>--thl-log-file-size</td>
<td>repl-thl-log-file-size</td>
<td>File size in bytes for THL disk logs</td>
</tr>
<tr>
<td>--thl-log-fsync</td>
<td>repl-thl-log-fsync</td>
<td>Fsync THL records on commit. More reliable operation but adds latency to replication when using low-performance storage</td>
</tr>
<tr>
<td>--repl-thl-log-fsync</td>
<td>repl-thl-log-fsync, thl-log-fsync</td>
<td></td>
</tr>
<tr>
<td>--thl-log-retention</td>
<td>repl-thl-log-retention</td>
<td>How long do you want to keep THL files.</td>
</tr>
<tr>
<td>--repl-thl-log-retention</td>
<td>repl-thl-log-retention, thl-log-retention</td>
<td></td>
</tr>
<tr>
<td>--thl-protocol</td>
<td>repl-thl-protocol</td>
<td>Protocol to use for THL communication with this service</td>
</tr>
<tr>
<td>--repl-thl-protocol</td>
<td>repl-thl-protocol, thl-protocol</td>
<td></td>
</tr>
<tr>
<td>--topology</td>
<td>dataservice-topology</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>--dataservice-topology</td>
<td>topology</td>
<td></td>
</tr>
<tr>
<td>--user</td>
<td>user</td>
<td>System User</td>
</tr>
<tr>
<td>--verbose</td>
<td>v</td>
<td>Display debug, info, notice, warning and error messages</td>
</tr>
<tr>
<td>-v</td>
<td>verbose</td>
<td></td>
</tr>
<tr>
<td>--vertica-dbname</td>
<td>repl-vertica-dbname</td>
<td>Name of the database to replicate into</td>
</tr>
<tr>
<td>--repl-vertica-dbname</td>
<td>repl-vertica-dbname, vertica-dbname</td>
<td></td>
</tr>
<tr>
<td>--witnesses</td>
<td>dataservice-witnesses</td>
<td>Witness hosts for the dataservice</td>
</tr>
<tr>
<td>--dataservice-witnesses</td>
<td>witness</td>
<td></td>
</tr>
</tbody>
</table>

---allow-bidi-unsafe---

<table>
<thead>
<tr>
<th>Option</th>
<th>--allow-bidi-unsafe [219]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-allow-bidi-unsafe [219]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>allow-bidi-unsafe [219], repl-allow-bidi-unsafe [219]</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, true</td>
</tr>
</tbody>
</table>

---api---

<table>
<thead>
<tr>
<th>Option</th>
<th>--api [219]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-api [219]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>api [219], repl-api [219]</td>
</tr>
<tr>
<td>Description</td>
<td>Enable the replication API</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

---

**--api-host**

<table>
<thead>
<tr>
<th>Option</th>
<th>--api-host [220]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Aliases</strong></td>
<td>--repl-api-host [220]</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>api-host [220], repl-api-host [220]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Hostname that the replication API should listen on</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

---

**--api-password**

<table>
<thead>
<tr>
<th>Option</th>
<th>--api-password [220]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Aliases</strong></td>
<td>--repl-api-password [220]</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>api-password [220], repl-api-password [220]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>HTTP basic auth password for the replication API</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

---

**--api-port**

<table>
<thead>
<tr>
<th>Option</th>
<th>--api-port [220]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Aliases</strong></td>
<td>--repl-api-port [220]</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>api-port [220], repl-api-port [220]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Port that the replication API should bind to</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

---

**--api-user**

<table>
<thead>
<tr>
<th>Option</th>
<th>--api-user [220]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Aliases</strong></td>
<td>--repl-api-user [220]</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>api-user [220], repl-api-user [220]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>HTTP basic auth username for the replication API</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

---

**--application-password**

<table>
<thead>
<tr>
<th>Option</th>
<th>--application-password [220]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Aliases</strong></td>
<td>--connector-password [220]</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>application-password [220], connector-password [220]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Database password for the connector</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

---

**--application-port**

<table>
<thead>
<tr>
<th>Option</th>
<th>--application-port [220]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Aliases</strong></td>
<td>--connector-listen-port [220]</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>application-port [220], connector-listen-port [220]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Port for the connector to listen on</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

---

**--application-readonly-port**

| Option                | --application-readonly-port [220] |
### Aliases

<table>
<thead>
<tr>
<th>Alias</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>--connector-readonly-listen-port</td>
<td>application-readonly-port, connector-readonly-listen-port</td>
</tr>
</tbody>
</table>

### Description

Port for the connector to listen for read-only connections on.

### Value Type

string

---

### Option: --application-user

<table>
<thead>
<tr>
<th>Alias</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>--connector-user</td>
<td>application-user</td>
</tr>
</tbody>
</table>

### Description

Database username for the connector.

### Value Type

string

---

### Option: --auto-enable

<table>
<thead>
<tr>
<th>Alias</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>--repl-auto-enable</td>
<td>auto-enable, repl-auto-enable</td>
</tr>
</tbody>
</table>

### Description

Auto-enable services after start-up.

### Value Type

string

---

### Option: --auto-recovery-delay-interval

<table>
<thead>
<tr>
<th>Config File Options</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto-recovery-delay-interval</td>
<td>auto-recovery-delay-interval</td>
</tr>
</tbody>
</table>

### Description

Delay between going OFFLINE and attempting to go ONLINE.

### Value Type

integer

### Default

5

The delay between the replicator identifying that autorecovery is needed, and autorecovery being attempted. For busy MySQL installations, larger numbers may be needed to allow time for MySQL servers to restart or recover from their failure.

---

### Option: --auto-recovery-max-attempts

<table>
<thead>
<tr>
<th>Config File Options</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto-recovery-max-attempts</td>
<td>auto-recovery-max-attempts</td>
</tr>
</tbody>
</table>

### Description

Maximum number of attempts at automatic recovery.

### Value Type

integer

### Default

0

Specifies the number of attempts the replicator will make to go back online. When the number of attempts has been reached, the replicator will remain in the OFFLINE 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.

---

### Option: --auto-recovery-reset-interval

<table>
<thead>
<tr>
<th>Config File Options</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto-recovery-reset-interval</td>
<td>auto-recovery-reset-interval</td>
</tr>
</tbody>
</table>

### Description

Delay before autorecovery is deemed to have succeeded.

### Value Type

integer

### Default

5

The time in ONLINE 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.
**Command-line Tools**

---

**--backup-directory**

<table>
<thead>
<tr>
<th>Option</th>
<th>--backup-directory [222]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-backup-directory [222]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>backup-directory [222], repl-backup-directory [222]</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>
</tbody>
</table>

**--backup-dump-directory**

<table>
<thead>
<tr>
<th>Option</th>
<th>--backup-dump-directory [222]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-backup-dump-directory [222]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>backup-dump-directory [222], repl-backup-dump-directory [222]</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 [222]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-backup-method [222]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>backup-method [222], repl-backup-method [222]</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>mysqldump</td>
</tr>
<tr>
<td></td>
<td>none</td>
</tr>
<tr>
<td></td>
<td>script</td>
</tr>
<tr>
<td></td>
<td>xtrabackup</td>
</tr>
<tr>
<td></td>
<td>xtrabackup-full</td>
</tr>
<tr>
<td></td>
<td>xtrabackup-incremental</td>
</tr>
</tbody>
</table>

**--backup-online**

<table>
<thead>
<tr>
<th>Option</th>
<th>--backup-online [222]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-backup-online [222]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>backup-online [222], repl-backup-online [222]</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 [222]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-backup-retention [222]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>backup-retention [222], repl-backup-retention [222]</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>

**--backup-script**

<table>
<thead>
<tr>
<th>Option</th>
<th>--backup-script [222]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-backup-script [222]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>backup-script [222], repl-backup-script [222]</td>
</tr>
<tr>
<td>Description</td>
<td>What is the path to the backup script</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>---------------------------------------</td>
</tr>
<tr>
<td>Value Type</td>
<td>filename</td>
</tr>
</tbody>
</table>

**--batch-enabled**

<table>
<thead>
<tr>
<th>Option</th>
<th>--batch-enabled [223]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>batch-enabled [223]</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 [223]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>batch-load-language [223]</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 [223]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>batch-load-template [223]</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>

**--channels**

<table>
<thead>
<tr>
<th>Option</th>
<th>--channels [223]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-channels [223]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>channels [223], repl-channels [223]</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>

**--composite-datasources**

<table>
<thead>
<tr>
<th>Option</th>
<th>--composite-datasources [223]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--dataservice-composite-datasources [223]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>composite-datasources [223], dataservice-composite-datasources [223]</td>
</tr>
<tr>
<td>Description</td>
<td>Data services that should be added to this composite data service</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

**--config-file-help**

<table>
<thead>
<tr>
<th>Option</th>
<th>--config-file-help [223]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>config-file-help [223]</td>
</tr>
<tr>
<td>Description</td>
<td>Display help information for content of the config file</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

**--conn-java-enable-concurrent-gc**

<table>
<thead>
<tr>
<th>Option</th>
<th>--conn-java-enable-concurrent-gc [223]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>conn-java-enable-concurrent-gc [223]</td>
</tr>
<tr>
<td>Description</td>
<td>Connector Java uses concurrent garbage collection</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td><strong>string</strong></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><strong>Option</strong></td>
<td><code>--conn-java-mem-size</code> [224]</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td><code>conn-java-mem-size</code> [224]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Connector Java heap memory size used to buffer data between clients and databases</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>numeric</td>
</tr>
<tr>
<td><strong>Default</strong></td>
<td>256</td>
</tr>
</tbody>
</table>

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.

| **Option** | `--conn-round-robin-include-master` [224] |
| **Config File Options** | `conn-round-robin-include-master` [224] |
| **Description** | Should the Connector include the master in round-robin load balancing |
| **Value Type** | string |

| **Option** | `--connector-autoreconnect` [224] |
| **Config File Options** | `connector-autoreconnect` [224] |
| **Description** | Enable auto-reconnect in the connector |
| **Value Type** | string |

| **Option** | `--connector-default-schema` [224] |
| **Aliases** | `--connector-forced-schema` [224] |
| **Config File Options** | `connector-default-schema` [224], `connector-forced-schema` [224] |
| **Description** | Default schema for the connector to use |
| **Value Type** | string |

| **Option** | `--connector-delete-user-map` [224] |
| **Config File Options** | `connector-delete-user-map` [224] |
| **Description** | Overwrite an existing user.map file |
| **Value Type** | string |

| **Option** | `--connector-drop-after-max-connections` [224] |
| **Config File Options** | `connector-drop-after-max-connections` [224] |
| **Description** | Instantly drop connections that arrive after `--connector-max-connections` has been reached |
| **Value Type** | string |

<p>| <strong>Option</strong> | <code>--connector-listen-interface</code> [224] |
| <strong>Config File Options</strong> | <code>connector-listen-interface</code> [224] |
| <strong>Description</strong> | Listen interface to use for the connector |</p>
<table>
<thead>
<tr>
<th>Value Type</th>
<th>string</th>
</tr>
</thead>
</table>

**--connector-max-connections**

<table>
<thead>
<tr>
<th>Option</th>
<th>--connector-max-connections [225]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Config File Options</strong></td>
<td>connector-max-connections [225]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>The maximum number of connections the connector should allow at any time</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>numeric</td>
</tr>
</tbody>
</table>

**--connector-max-slave-latency**

<table>
<thead>
<tr>
<th>Option</th>
<th>--connector-max-slave-latency [225]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Aliases</strong></td>
<td>--connector-max-applied-latency [225]</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>connector-max-applied-latency [225], connector-max-slave-latency [225]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>The maximum applied latency for slave connections</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

**--connector-readonly**

<table>
<thead>
<tr>
<th>Option</th>
<th>--connector-readonly [225]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Aliases</strong></td>
<td>--enable-connector-readonly [225]</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>connector-readonly [225], enable-connector-readonly [225]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Enable the Tungsten Connector read-only mode</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

**--connector-ro-addresses**

<table>
<thead>
<tr>
<th>Option</th>
<th>--connector-ro-addresses [225]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Config File Options</strong></td>
<td>connector-ro-addresses [225]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Connector addresses that should receive a r/o connection</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

**--connector-rw-addresses**

<table>
<thead>
<tr>
<th>Option</th>
<th>--connector-rw-addresses [225]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Config File Options</strong></td>
<td>connector-rw-addresses [225]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Connector addresses that should receive a r/w connection</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

**--connector-rwsplitting**

<table>
<thead>
<tr>
<th>Option</th>
<th>--connector-rwsplitting [225]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Config File Options</strong></td>
<td>connector-rwsplitting [225]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Enable DirectReads R/W splitting in the connector</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

**--connector-smartscale**

<table>
<thead>
<tr>
<th>Option</th>
<th>--connector-smartscale [225]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Config File Options</strong></td>
<td>connector-smartscale [225]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Enable SmartScale R/W splitting in the connector</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

**--connector-smartscale-sessionid**
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>--connector-smartscale-sessionid</td>
<td>The default session ID to use with smart scale</td>
<td>string</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>connector-smartscale-sessionid [225]</td>
<td></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>--connectors</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Option</strong></td>
<td>--connectors [226]</td>
<td></td>
</tr>
<tr>
<td><strong>Aliases</strong></td>
<td>--dataservice-connectors [226]</td>
<td></td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>connectors [226], dataservice-connectors [226]</td>
<td></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Hostnames for the dataservice connectors</td>
<td>string</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>--consistency-policy</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Option</strong></td>
<td>--consistency-policy [226]</td>
<td></td>
</tr>
<tr>
<td><strong>Aliases</strong></td>
<td>--repl-consistency-policy [226]</td>
<td></td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>consistency-policy [226], repl-consistency-policy [226]</td>
<td></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Should the replicator stop or warn if a consistency check fails?</td>
<td></td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>--dataservice-name</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Option</strong></td>
<td>--dataservice-name [226]</td>
<td></td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>dataservice-name [226]</td>
<td></td>
</tr>
<tr>
<td><strong>Description</strong></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></td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td></td>
<td>string</td>
</tr>
<tr>
<td><strong>--dataservice-relay-enabled</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Option</strong></td>
<td>--dataservice-relay-enabled [226]</td>
<td></td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>dataservice-relay-enabled [226]</td>
<td></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Make this dataservice the slave of another</td>
<td>string</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>--dataservice-schema</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Option</strong></td>
<td>--dataservice-schema [226]</td>
<td></td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>dataservice-schema [226]</td>
<td></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>The db schema to hold dataservice details</td>
<td>string</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>--dataservice-thl-port</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Option</strong></td>
<td>--dataservice-thl-port [226]</td>
<td></td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>dataservice-thl-port [226]</td>
<td></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Port to use for THL operations</td>
<td>string</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>--dataservice-vip-enabled</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Option</strong></td>
<td>--dataservice-vip-enabled [226]</td>
<td></td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>dataservice-vip-enabled [226]</td>
<td></td>
</tr>
</tbody>
</table>
## Command-line Tools

<table>
<thead>
<tr>
<th>Description</th>
<th>Is VIP management enabled?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Value Type</strong></td>
<td><strong>string</strong></td>
</tr>
</tbody>
</table>

### --dataservice-vip-ipaddress

<table>
<thead>
<tr>
<th>Option</th>
<th>--dataservice-vip-ipaddress [227]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Config File Options</strong></td>
<td>dataservice-vip-ipaddress [227]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>VIP IP address</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td><strong>string</strong></td>
</tr>
</tbody>
</table>

### --dataservice-vip-netmask

<table>
<thead>
<tr>
<th>Option</th>
<th>--dataservice-vip-netmask [227]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Config File Options</strong></td>
<td>dataservice-vip-netmask [227]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>VIP netmask</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td><strong>string</strong></td>
</tr>
</tbody>
</table>

### --datasource-boot-script

<table>
<thead>
<tr>
<th>Option</th>
<th>--datasource-boot-script [227]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Aliases</strong></td>
<td>--repl-datasource-boot-script [227]</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>datasource-boot-script [227], repl-datasource-boot-script [227]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Database start script</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td><strong>string</strong></td>
</tr>
</tbody>
</table>

### --datasource-log-directory

<table>
<thead>
<tr>
<th>Option</th>
<th>--datasource-log-directory [227]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Aliases</strong></td>
<td>--repl-datasource-log-directory [227]</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>datasource-log-directory [227], repl-datasource-log-directory [227]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Master log directory</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td><strong>string</strong></td>
</tr>
</tbody>
</table>

### --datasource-log-pattern

<table>
<thead>
<tr>
<th>Option</th>
<th>--datasource-log-pattern [227]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Aliases</strong></td>
<td>--repl-datasource-log-pattern [227]</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>datasource-log-pattern [227], repl-datasource-log-pattern [227]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Master log filename pattern</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td><strong>string</strong></td>
</tr>
</tbody>
</table>

### --datasource-mysql-conf

<table>
<thead>
<tr>
<th>Option</th>
<th>--datasource-mysql-conf [227]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Aliases</strong></td>
<td>--repl-datasource-mysql-conf [227]</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>datasource-mysql-conf [227], repl-datasource-mysql-conf [227]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>MySQL config file</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td><strong>string</strong></td>
</tr>
</tbody>
</table>

### --datasource-mysql-data-directory

<table>
<thead>
<tr>
<th>Option</th>
<th>--datasource-mysql-data-directory [227]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Aliases</strong></td>
<td>--repl-datasource-mysql-data-directory [227]</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>datasource-mysql-data-directory [227], repl-datasource-mysql-data-directory [227]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td><strong>MySQL data directory</strong></td>
</tr>
<tr>
<td>----------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th><strong>Option</strong></th>
<th><code>--datasource-mysql-ibdata-directory [228]</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Aliases</strong></td>
<td><code>--repl-datasource-mysql-ibdata-directory [228]</code></td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td><code>datasource-mysql-ibdata-directory [228], repl-datasource-mysql-ibdata-directory [228]</code></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>MySQL InnoDB data directory</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th><strong>Option</strong></th>
<th><code>--datasource-mysql-iblog-directory [228]</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Aliases</strong></td>
<td><code>--repl-datasource-mysql-iblog-directory [228]</code></td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td><code>datasource-mysql-iblog-directory [228], repl-datasource-mysql-iblog-directory [228]</code></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>MySQL InnoDB log directory</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th><strong>Option</strong></th>
<th><code>--datasource-oracle-scan [228]</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Aliases</strong></td>
<td><code>--repl-datasource-oracle-scan [228]</code></td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td><code>datasource-oracle-scan [228], repl-datasource-oracle-scan [228]</code></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Oracle SCAN</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th><strong>Option</strong></th>
<th><code>--datasource-oracle-service [228]</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Aliases</strong></td>
<td><code>--repl-datasource-oracle-service [228]</code></td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td><code>datasource-oracle-service [228], repl-datasource-oracle-service [228]</code></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Oracle Service</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th><strong>Option</strong></th>
<th><code>--datasource-pg-archive [228]</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Aliases</strong></td>
<td><code>--repl-datasource-pg-archive [228]</code></td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td><code>datasource-pg-archive [228], repl-datasource-pg-archive [228]</code></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>PostgreSQL archive location</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th><strong>Option</strong></th>
<th><code>--datasource-pg-conf [228]</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Aliases</strong></td>
<td><code>--repl-datasource-pg-conf [228]</code></td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td><code>datasource-pg-conf [228], repl-datasource-pg-conf [228]</code></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Location of postgresql.conf</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th><strong>Option</strong></th>
<th><code>--datasource-pg-home [228]</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Command-line Tools</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td></td>
</tr>
</tbody>
</table>

### Aliases
- `--repl-datasource-pg-home [228]`

### Config File Options
- `datasource-pg-home [228], repl-datasource-pg-home [228]`

### Description
PostgreSQL data directory

### Value Type
string

---

### `--datasource-pg-root`

<table>
<thead>
<tr>
<th>Option</th>
<th><code>--datasource-pg-root [229]</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td><code>--repl-datasource-pg-root [229]</code></td>
</tr>
<tr>
<td>Config File Options</td>
<td><code>datasource-pg-root [229], repl-datasource-pg-root [229]</code></td>
</tr>
</tbody>
</table>

### Description
Root directory for postgresql installation

### Value Type
string

---

### `--datasource-type`

<table>
<thead>
<tr>
<th>Option</th>
<th><code>--datasource-type [229]</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td><code>--repl-datasource-type [229]</code></td>
</tr>
<tr>
<td>Config File Options</td>
<td><code>datasource-type [229], repl-datasource-type [229]</code></td>
</tr>
</tbody>
</table>

### Description
Database type

### Value Type
string

<table>
<thead>
<tr>
<th>Default</th>
<th>mysql</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Valid Values</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td>mongodb</td>
<td>MongoDB</td>
</tr>
<tr>
<td>mysql</td>
<td>MySQL</td>
</tr>
<tr>
<td>oracle</td>
<td>Oracle</td>
</tr>
<tr>
<td>postgresql</td>
<td>PostgreSQL</td>
</tr>
<tr>
<td>postgresql-wal</td>
<td>PostgreSQL (using Write Ahead Logging)</td>
</tr>
<tr>
<td>vertica</td>
<td>Vertica</td>
</tr>
</tbody>
</table>

---

### `--delete`

<table>
<thead>
<tr>
<th>Option</th>
<th><code>--delete [229]</code></th>
</tr>
</thead>
</table>

| Config File Options | delete [229] |

### Description
Delete the named data service from the configuration

<table>
<thead>
<tr>
<th>Data Service options:</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
</tr>
</tbody>
</table>

---

### `--deploy-current-package`

<table>
<thead>
<tr>
<th>Option</th>
<th><code>--deploy-current-package [229]</code></th>
</tr>
</thead>
</table>

| Config File Options | deploy-current-package [229] |

### Description
Deploy the current Tungsten package

### Value Type
string

---

### `--deploy-package-uri`

<table>
<thead>
<tr>
<th>Option</th>
<th><code>--deploy-package-uri [229]</code></th>
</tr>
</thead>
</table>

| Config File Options | deploy-package-uri [229] |

### Description
URL for the Tungsten package to deploy

### Value Type
string

---

### `--direct-datasource-host`

<table>
<thead>
<tr>
<th>Option</th>
<th><code>--direct-datasource-host [229]</code></th>
</tr>
</thead>
</table>

<p>| Aliases        | <code>--repl-direct-datasource-host [229]</code> |</p>
<table>
<thead>
<tr>
<th>Config File Options</th>
<th>direct-datasource-host [229], repl-direct-datasource-host [229]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Database server hostname</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

---

**--direct-datasource-log-directory**

<table>
<thead>
<tr>
<th>Option</th>
<th>--direct-datasource-log-directory [230]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-direct-datasource-log-directory [230]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>direct-datasource-log-directory [230], repl-direct-datasource-log-directory [230]</td>
</tr>
<tr>
<td>Description</td>
<td>Master log directory</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

---

**--direct-datasource-log-pattern**

<table>
<thead>
<tr>
<th>Option</th>
<th>--direct-datasource-log-pattern [230]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-direct-datasource-log-pattern [230]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>direct-datasource-log-pattern [230], repl-direct-datasource-log-pattern [230]</td>
</tr>
<tr>
<td>Description</td>
<td>Master log filename pattern</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

---

**--direct-datasource-oracle-scan**

<table>
<thead>
<tr>
<th>Option</th>
<th>--direct-datasource-oracle-scan [230]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-direct-datasource-oracle-scan [230]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>direct-datasource-oracle-scan [230], repl-direct-datasource-oracle-scan [230]</td>
</tr>
<tr>
<td>Description</td>
<td>Oracle SCAN</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

---

**--direct-datasource-oracle-service**

<table>
<thead>
<tr>
<th>Option</th>
<th>--direct-datasource-oracle-service [230]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-direct-datasource-oracle-service [230]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>direct-datasource-oracle-service [230], repl-direct-datasource-oracle-service [230]</td>
</tr>
<tr>
<td>Description</td>
<td>Oracle Service</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

---

**--direct-datasource-port**

<table>
<thead>
<tr>
<th>Option</th>
<th>--direct-datasource-port [230]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-direct-datasource-port [230]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>direct-datasource-port [230], repl-direct-datasource-port [230]</td>
</tr>
<tr>
<td>Description</td>
<td>Database server port</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

---

**--direct-datasource-type**

<table>
<thead>
<tr>
<th>Option</th>
<th>--direct-datasource-type [230]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-direct-datasource-type [230]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>direct-datasource-type [230], repl-direct-datasource-type [230]</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>mongodb, MongoDB</td>
</tr>
</tbody>
</table>
### Command-line Tools

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>mysql</td>
<td>MySQL (using Write Ahead Logging)</td>
<td></td>
</tr>
<tr>
<td>oracle</td>
<td>Oracle</td>
<td></td>
</tr>
<tr>
<td>postgresql</td>
<td>PostgreSQL</td>
<td></td>
</tr>
<tr>
<td>postgresql-wal</td>
<td>PostgreSQL (using Write Ahead Logging)</td>
<td></td>
</tr>
<tr>
<td>vertica</td>
<td>Vertica</td>
<td></td>
</tr>
</tbody>
</table>

#### --direct-replication-password

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>direct-replication-password</td>
<td>Database password</td>
<td>string</td>
</tr>
</tbody>
</table>

#### --direct-replication-user

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>direct-replication-user</td>
<td>Database login for Tungsten</td>
<td>string</td>
</tr>
</tbody>
</table>

#### --disable-relay-logs

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>disable-relay-logs</td>
<td>Disable the use of relay-logs?</td>
<td>string</td>
</tr>
</tbody>
</table>

#### --enable-active-witnesses

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable-active-witnesses</td>
<td>Enable active witness hosts</td>
<td>boolean</td>
</tr>
</tbody>
</table>

#### --enable-connector-bridge-mode

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable-connector-bridge-mode</td>
<td>Enable the Tungsten Connector bridge mode</td>
<td>string</td>
</tr>
</tbody>
</table>

#### --enable-connector-ssl

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable-connector-ssl</td>
<td>Enable SSL encryption of connector traffic to the database</td>
<td>string</td>
</tr>
</tbody>
</table>
### Command-line Tools

<table>
<thead>
<tr>
<th>Value Type</th>
<th>string</th>
</tr>
</thead>
</table>

#### --enable-heterogenous-master

<table>
<thead>
<tr>
<th>Option</th>
<th>--enable-heterogenous-master [232]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>enable-heterogenous-master [232]</td>
</tr>
<tr>
<td>Description</td>
<td>Enable heterogenous operation for the master</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### --enable-heterogenous-service

<table>
<thead>
<tr>
<th>Option</th>
<th>--enable-heterogenous-service [232]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>enable-heterogenous-service [232]</td>
</tr>
<tr>
<td>Description</td>
<td>Enable heterogenous operation</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

- **On a Master**
  - `--mysql-use-bytes-for-string [240]` 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 [240]` is set to true.
  - `pkey` filter is enabled (in the `q-to-dbms` stage).

#### --enable-heterogenous-slave

<table>
<thead>
<tr>
<th>Option</th>
<th>--enable-heterogenous-slave [232]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>enable-heterogenous-slave [232]</td>
</tr>
<tr>
<td>Description</td>
<td>Enable heterogenous operation for the slave</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### --enable-rmi-authentication

<table>
<thead>
<tr>
<th>Option</th>
<th>--enable-rmi-authentication [232]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--rmi-authentication [232]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>enable-rmi-authentication [232], rmi-authentication [232]</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 [232]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--rmi-ssl [232]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>enable-rmi-ssl [232], rmi-ssl [232]</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
### Command-line Tools

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--enable-slave-thl-listener</code></td>
<td>Should this service allow THL connections?</td>
<td>string</td>
</tr>
<tr>
<td><code>--repl-enable-slave-thl-listener</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>enable-slave-thl-listener</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>repl-enable-slave-thl-listener</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--enable-sudo-access</code></td>
<td>Run root commands using sudo</td>
<td>string</td>
</tr>
<tr>
<td><code>--root-command-prefix</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>enable-sudo-access</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>root-command-prefix</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--enable-thl-ssl</code></td>
<td>Enable SSL encryption of THL communication for this service</td>
<td>string</td>
</tr>
<tr>
<td><code>--repl-enable-thl-ssl</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>--thl-ssl</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>enable-thl-ssl</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>repl-enable-thl-ssl</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>thl-ssl</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--enable-validation-check String</code></td>
<td>Remove a corresponding --skip-validation-check argument</td>
<td>string</td>
</tr>
<tr>
<td><code>enable-validation-check String</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--enable-validation-warnings String</code></td>
<td>Remove a corresponding --skip-validation-warnings argument</td>
<td>string</td>
</tr>
<tr>
<td><code>enable-validation-warnings String</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--force</code></td>
<td>Do not display confirmation prompts or stop the configure process for errors</td>
<td>string</td>
</tr>
<tr>
<td><code>-r</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>r</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>force</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--help</code></td>
<td>Displays help message</td>
<td>string</td>
</tr>
<tr>
<td><code>-h</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>h</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--host-name</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Command-line Tools</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>host-name [233]</td>
<td></td>
</tr>
<tr>
<td>Config File Options</td>
<td>host-name [233]</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>DNS hostname</td>
<td></td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

--hosts

| Option | hosts [234] |
| Config File Options | hosts [234] |
| 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 [234] |
| Aliases | dataservice-hub-host [234] |
| Config File Options | dataservice-hub-host [234], hub [234] |
| Description | What is the hub host for this all-masters dataservice? |
| Value Type | string |

--hub-service

| Option | hub-service [234] |
| Aliases | dataservice-hub-service [234] |
| Config File Options | dataservice-hub-service [234], hub-service [234] |
| Description | The data service to use for the hub of a star topology |
| Value Type | string |

--info

| Option | info [234] |
| Aliases | -i [234] |
| Config File Options | i [234], info [234] |
| Description | Display info, notice, warning and error messages |
| Value Type | string |

--install

| Option | install [234] |
| Config File Options | install [234] |
| Description | Install service start scripts |
| Value Type | string |

--install-directory

| Option | install-directory [234] |
| Aliases | home-directory [234] |
| Config File Options | home-directory [234], install-directory [234] |
| Description | Installation directory |
| Value Type | string |

--java-connector-keystore-password

| Option | java-connector-keystore-password [234] |
## Command-line Tools

<table>
<thead>
<tr>
<th>Config File Options</th>
<th>java-connector-keystore-password [234]</th>
</tr>
</thead>
<tbody>
<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>

---

**Option**  
```
--java-connector-keystore-path [235]
```

<table>
<thead>
<tr>
<th>Config File Options</th>
<th>java-connector-keystore-path [235]</th>
</tr>
</thead>
<tbody>
<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>

---

**Option**  
```
--java-connector-truststore-password [235]
```

<table>
<thead>
<tr>
<th>Config File Options</th>
<th>java-connector-truststore-password [235]</th>
</tr>
</thead>
<tbody>
<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>

---

**Option**  
```
--java-connector-truststore-path [235]
```

<table>
<thead>
<tr>
<th>Config File Options</th>
<th>java-connector-truststore-path [235]</th>
</tr>
</thead>
<tbody>
<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>

---

**Option**  
```
--java-enable-concurrent-gc [235]
```

<table>
<thead>
<tr>
<th>Aliases</th>
<th>--repl-java-enable-concurrent-gc [235]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>java-enable-concurrent-gc [235], repl-java-enable-concurrent-gc [235]</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>

---

**Option**  
```
--java-file-encoding [235]
```

<table>
<thead>
<tr>
<th>Aliases</th>
<th>--repl-java-file-encoding [235]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>java-file-encoding [235], repl-java-file-encoding [235]</td>
</tr>
<tr>
<td>Description</td>
<td>Java platform charset (esp. for heterogeneous replication)</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

---

**Option**  
```
--java-jmxremote-access-path [235]
```

<table>
<thead>
<tr>
<th>Config File Options</th>
<th>java-jmxremote-access-path [235]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Local path to the Java JMX Remote Access file.</td>
</tr>
<tr>
<td>Value Type</td>
<td>filename</td>
</tr>
</tbody>
</table>

---

**Option**  
```
--java-keystore-password [235]
```

<table>
<thead>
<tr>
<th>Config File Options</th>
<th>java-keystore-password [235]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The password for unlocking the tungsten_keystore.jks file in the security directory</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>
### Command-line Tools

#### --java-keystore-path

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>--java-keystore-path</td>
<td>[236]</td>
</tr>
</tbody>
</table>

**Config File Options**

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>java-keystore-path [236]</td>
</tr>
</tbody>
</table>

**Description**

Local path to the Java Keystore file.

**Value Type**

filename

#### --java-mem-size

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>--java-mem-size</td>
<td>[236]</td>
</tr>
</tbody>
</table>

**Aliases**

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>--repl-java-mem-size [236]</td>
</tr>
</tbody>
</table>

**Config File Options**

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>java-mem-size [236], repl-java-mem-size [236]</td>
</tr>
</tbody>
</table>

**Description**

Replicator Java heap memory size in Mb (min 128)

**Value Type**

numeric

#### --java-passwordstore-path

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>--java-passwordstore-path</td>
<td>[236]</td>
</tr>
</tbody>
</table>

**Config File Options**

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>java-passwordstore-path [236]</td>
</tr>
</tbody>
</table>

**Description**

Local path to the Java Password Store file.

**Value Type**

filename

#### --java-truststore-password

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>--java-truststore-password</td>
<td>[236]</td>
</tr>
</tbody>
</table>

**Config File Options**

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>java-truststore-password [236]</td>
</tr>
</tbody>
</table>

**Description**

The password for unlocking the tungsten_truststore.jks file in the security directory

**Value Type**

string

#### --java-truststore-path

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>--java-truststore-path</td>
<td>[236]</td>
</tr>
</tbody>
</table>

**Config File Options**

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>java-truststore-path [236]</td>
</tr>
</tbody>
</table>

**Description**

Local path to the Java Truststore file.

**Value Type**

filename

#### --java-user-timezone

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>--java-user-timezone</td>
<td>[236]</td>
</tr>
</tbody>
</table>

**Aliases**

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>--repl-java-user-timezone [236]</td>
</tr>
</tbody>
</table>

**Config File Options**

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>java-user-timezone [236], repl-java-user-timezone [236]</td>
</tr>
</tbody>
</table>

**Description**

Java VM Timezone (esp. for cross-site replication)

**Value Type**

numeric

#### --log

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>--log</td>
<td>[236]</td>
</tr>
</tbody>
</table>

**Config File Options**

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>log [236]</td>
</tr>
</tbody>
</table>

**Description**

Write all messages, visible and hidden, to this file. You may specify a filename, ’pid’ or ’timestamp’.

**Value Type**

numeric

#### --log-slave-updates

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>--log-slave-updates</td>
<td>[236]</td>
</tr>
</tbody>
</table>
### Command-line Tools

#### Config File Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>log-slave-updates</code></td>
<td>string</td>
<td>Should slaves log updates to binlog</td>
</tr>
</tbody>
</table>

---

#### --master

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--master</code></td>
<td>string</td>
<td>--master [237]</td>
</tr>
<tr>
<td><code>--dataservice-master-host</code></td>
<td>string</td>
<td>--dataservice-master-host [237], --masters [237]</td>
</tr>
<tr>
<td><code>master</code></td>
<td>string</td>
<td>dataservice-master-host [237], master [237], masters [237]</td>
</tr>
<tr>
<td><code>masters</code></td>
<td>string</td>
<td>What is the master host for this dataservice?</td>
</tr>
</tbody>
</table>

---

#### --master-preferred-role

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--master-preferred-role</code></td>
<td>string</td>
<td>--master-preferred-role [237]</td>
</tr>
<tr>
<td><code>repl-master-preferred-role</code></td>
<td>string</td>
<td>--repl-master-preferred-role [237]</td>
</tr>
<tr>
<td><code>master-preferred-role</code></td>
<td>string</td>
<td>master-preferred-role [237], repl-master-preferred-role [237]</td>
</tr>
<tr>
<td><code>repl-master-preferred-role</code></td>
<td>string</td>
<td>Preferred role for master THL when connecting as a slave (master, slave, etc.)</td>
</tr>
</tbody>
</table>

---

#### --master-services

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--master-services</code></td>
<td>string</td>
<td>--master-services [237]</td>
</tr>
<tr>
<td><code>--dataservice-master-services</code></td>
<td>string</td>
<td>--dataservice-master-services [237]</td>
</tr>
<tr>
<td><code>dataservice-master-services</code></td>
<td>string</td>
<td>dataservice-master-services [237], master-services [237]</td>
</tr>
<tr>
<td><code>master-services</code></td>
<td>string</td>
<td>Data service names that should be used on each master</td>
</tr>
</tbody>
</table>

---

#### --members

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--members</code></td>
<td>string</td>
<td>--members [237]</td>
</tr>
<tr>
<td><code>--dataservice-hosts</code></td>
<td>string</td>
<td>--dataservice-hosts [237]</td>
</tr>
<tr>
<td><code>dataservice-hosts</code></td>
<td>string</td>
<td>dataservice-hosts [237], members [237]</td>
</tr>
<tr>
<td><code>members</code></td>
<td>string</td>
<td>Hostnames for the dataservice members</td>
</tr>
</tbody>
</table>

---

#### --mgr-api

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--mgr-api</code></td>
<td>string</td>
<td>--mgr-api [237]</td>
</tr>
<tr>
<td><code>mgr-api</code></td>
<td>string</td>
<td>mgr-api [237]</td>
</tr>
<tr>
<td><code>mgr-api</code></td>
<td>string</td>
<td>Enable the Manager API</td>
</tr>
</tbody>
</table>

---

#### --mgr-api-address

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--mgr-api-address</code></td>
<td>string</td>
<td>--mgr-api-address [237]</td>
</tr>
<tr>
<td><code>mgr-api-address</code></td>
<td>string</td>
<td>mgr-api-address [237]</td>
</tr>
<tr>
<td><code>mgr-api-address</code></td>
<td>string</td>
<td>Address for the Manager API</td>
</tr>
</tbody>
</table>

---

#### --mgr-api-port

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--mgr-api-port</code></td>
<td>string</td>
<td>--mgr-api-port [237]</td>
</tr>
<tr>
<td><code>mgr-api-port</code></td>
<td>string</td>
<td>mgr-api-port [237]</td>
</tr>
<tr>
<td>Description</td>
<td>Port for the Manager API</td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------------</td>
<td></td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

`--mgr-group-communication-port`

<table>
<thead>
<tr>
<th>Option</th>
<th><code>--mgr-group-communication-port</code> [238]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td><code>mgr-group-communication-port</code> [238]</td>
</tr>
<tr>
<td>Description</td>
<td>Port to use for manager group communication</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

`--mgr-java-enable-concurrent-gc`

<table>
<thead>
<tr>
<th>Option</th>
<th><code>--mgr-java-enable-concurrent-gc</code> [238]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td><code>mgr-java-enable-concurrent-gc</code> [238]</td>
</tr>
<tr>
<td>Description</td>
<td>Manager Java uses concurrent garbage collection</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

`--mgr-java-mem-size`

<table>
<thead>
<tr>
<th>Option</th>
<th><code>--mgr-java-mem-size</code> [238]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td><code>mgr-java-mem-size</code> [238]</td>
</tr>
<tr>
<td>Description</td>
<td>Manager Java heap memory size in Mb (min 128)</td>
</tr>
<tr>
<td>Value Type</td>
<td>numeric</td>
</tr>
</tbody>
</table>

`--mgr-listen-interface`

<table>
<thead>
<tr>
<th>Option</th>
<th><code>--mgr-listen-interface</code> [238]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td><code>mgr-listen-interface</code> [238]</td>
</tr>
<tr>
<td>Description</td>
<td>Listen interface to use for the manager</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

`--mgr-policy-mode`

<table>
<thead>
<tr>
<th>Option</th>
<th><code>--mgr-policy-mode</code> [238]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td><code>mgr-policy-mode</code> [238]</td>
</tr>
<tr>
<td>Description</td>
<td>Manager policy mode</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
<tr>
<td>Valid Values</td>
<td>automatic</td>
</tr>
<tr>
<td></td>
<td>maintenance</td>
</tr>
<tr>
<td></td>
<td>manual</td>
</tr>
</tbody>
</table>

`--mgr-rmi-port`

<table>
<thead>
<tr>
<th>Option</th>
<th><code>--mgr-rmi-port</code> [238]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td><code>mgr-rmi-port</code> [238]</td>
</tr>
<tr>
<td>Description</td>
<td>Port to use for the manager RMI server</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

`--mgr-rmi-remote-port`

<table>
<thead>
<tr>
<th>Option</th>
<th><code>--mgr-rmi-remote-port</code> [238]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td><code>mgr-rmi-remote-port</code> [238]</td>
</tr>
<tr>
<td>Description</td>
<td>Port to use for calling the remote manager RMI server</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
<tr>
<td>---------------</td>
<td>----------------</td>
</tr>
</tbody>
</table>

**--mgr-ro-slave**

<table>
<thead>
<tr>
<th>Option</th>
<th>--mgr-ro-slave [239]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>mgr-ro-slave [239]</td>
</tr>
<tr>
<td>Description</td>
<td>Make slaves read-only</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

**--mgr-vip-arp-path**

<table>
<thead>
<tr>
<th>Option</th>
<th>--mgr-vip-arp-path [239]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>mgr-vip-arp-path [239]</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-vip-device**

<table>
<thead>
<tr>
<th>Option</th>
<th>--mgr-vip-device [239]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>mgr-vip-device [239]</td>
</tr>
<tr>
<td>Description</td>
<td>VIP network device</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

**--mgr-vip-ifconfig-path**

<table>
<thead>
<tr>
<th>Option</th>
<th>--mgr-vip-ifconfig-path [239]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>mgr-vip-ifconfig-path [239]</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 [239]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>mgr-wait-for-members [239]</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-connectorj-path**

<table>
<thead>
<tr>
<th>Option</th>
<th>--mysql-connectorj-path [239]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>mysql-connectorj-path [239]</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>

**--mysql-driver**

<table>
<thead>
<tr>
<th>Option</th>
<th>--mysql-driver [239]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>mysql-driver [239]</td>
</tr>
<tr>
<td>Description</td>
<td>MySQL Driver Vendor</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

**--mysql-enable-ansiquotes**

<table>
<thead>
<tr>
<th>Option</th>
<th>--mysql-enable-ansiquotes [239]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>repl-mysql-enable-ansiquotes [239]</td>
</tr>
</tbody>
</table>
### Command-line Tools

<table>
<thead>
<tr>
<th>Config File Options</th>
<th>mysql-enable-ansiquotes [239], repl-mysql-enable-ansiquotes [239]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Enables ANSI_QUOTES mode for incoming events?</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

---

**--mysql-enable-noonlykeywords**

<table>
<thead>
<tr>
<th>Option</th>
<th>--mysql-enable-noonlykeywords [240]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-mysql-enable-noonlykeywords [240]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>mysql-enable-noonlykeywords [240], repl-mysql-enable-noonlykeywords [240]</td>
</tr>
<tr>
<td>Description</td>
<td>Translates DELETE FROM ONLY } DELETE FROM and UPDATE ONLY } UPDATE.</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

---

**--mysql-enable-settostring**

<table>
<thead>
<tr>
<th>Option</th>
<th>--mysql-enable-settostring [240]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-mysql-enable-settostring [240]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>mysql-enable-settostring [240], repl-mysql-enable-settostring [240]</td>
</tr>
<tr>
<td>Description</td>
<td>Decode SET values into their text values?</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

---

**--mysql-ro-slave**

<table>
<thead>
<tr>
<th>Option</th>
<th>--mysql-ro-slave [240]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-mysql-ro-slave [240]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>mysql-ro-slave [240], repl-mysql-ro-slave [240]</td>
</tr>
<tr>
<td>Description</td>
<td>Slaves are read-only?</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

---

**--mysql-server-id**

<table>
<thead>
<tr>
<th>Option</th>
<th>--mysql-server-id [240]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-mysql-server-id [240]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>mysql-server-id [240], repl-mysql-server-id [240]</td>
</tr>
<tr>
<td>Description</td>
<td>Explicitly set the MySQL server ID</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>Option</th>
<th>--mysql-use-bytes-for-string [240]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-mysql-use-bytes-for-string [240]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>mysql-use-bytes-for-string [240], repl-mysql-use-bytes-for-string [240]</td>
</tr>
<tr>
<td>Description</td>
<td>Transfer strings as their byte representation?</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

---

**--mysql-xtrabackup-dir**

<table>
<thead>
<tr>
<th>Option</th>
<th>--mysql-xtrabackup-dir [240]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-mysql-xtrabackup-dir [240]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>mysql-xtrabackup-dir [240], repl-mysql-xtrabackup-dir [240]</td>
</tr>
<tr>
<td>Description</td>
<td>Directory to use for storing xtrabackup full &amp; incremental backups</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
<tr>
<td>------------------</td>
<td>--------</td>
</tr>
<tr>
<td><strong>--native-slave-takeover</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Option</strong></td>
<td><strong>--native-slave-takeover</strong> [241]</td>
</tr>
<tr>
<td><strong>Aliases</strong></td>
<td><strong>--repl-native-slave-takeover</strong> [241]</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>native-slave-takeover [241], repl-native-slave-takeover [241]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Take over native replication</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
<tr>
<td><strong>--net-ssh-option=key=value</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Option</strong></td>
<td><strong>--net-ssh-option=key=value</strong> [241]</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>net-ssh-option=key=value [241]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Set the Net::SSH option for remote system calls</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
<tr>
<td><strong>--no-deployment</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Option</strong></td>
<td><strong>--no-deployment</strong> [241]</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>no-deployment [241]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Skip deployment steps that create the install directory</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
<tr>
<td><strong>--no-validation</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Option</strong></td>
<td><strong>--no-validation</strong> [241]</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>no-validation [241]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Skip validation checks that run on each host</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
<tr>
<td><strong>--notice</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Option</strong></td>
<td><strong>--notice</strong> [241]</td>
</tr>
<tr>
<td><strong>Aliases</strong></td>
<td><strong>-n</strong> [241]</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>n [241], notice [241]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Display notice, warning and error messages</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>string</td>
</tr>
<tr>
<td><strong>--pg-archive-timeout</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Option</strong></td>
<td><strong>--pg-archive-timeout</strong> [241]</td>
</tr>
<tr>
<td><strong>Aliases</strong></td>
<td><strong>--repl-pg-archive-timeout</strong> [241]</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>pg-archive-timeout [241], repl-pg-archive-timeout [241]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Timeout for sending unfilled WAL buffers (data loss window)</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>numeric</td>
</tr>
<tr>
<td><strong>--pg-ctl</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Option</strong></td>
<td><strong>--pg-ctl</strong> [241]</td>
</tr>
<tr>
<td><strong>Aliases</strong></td>
<td><strong>--repl-pg-ctl</strong> [241]</td>
</tr>
<tr>
<td><strong>Config File Options</strong></td>
<td>pg-ctl [241], repl-pg-ctl [241]</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Path to the pg_ctl script</td>
</tr>
<tr>
<td><strong>Value Type</strong></td>
<td>filename</td>
</tr>
</tbody>
</table>
## Command-line Tools

### --pg-method

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>--pg-method</td>
<td>string</td>
</tr>
</tbody>
</table>

### --pg-standby

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>--pg-standby</td>
<td>filename</td>
</tr>
</tbody>
</table>

### --postgresql-dbname

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>--postgresql-dbname</td>
<td>string</td>
</tr>
</tbody>
</table>

### --postgresql-enable-mysql2pgddl

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>--postgresql-enable-mysql2pgddl</td>
<td>boolean</td>
</tr>
</tbody>
</table>

### --postgresql-slonik

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>--postgresql-slonik</td>
<td>filename</td>
</tr>
</tbody>
</table>

### --postgresql-tables

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>--postgresql-tables</td>
<td>string</td>
</tr>
</tbody>
</table>

### --preferred-path

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>--preferred-path</td>
<td>string</td>
</tr>
</tbody>
</table>
Table: Command-line tools

<table>
<thead>
<tr>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Additional command path</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 Continuent Tungsten, 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
shell> tpm ... --preferred-path=/usr/local/bin:/opt/bin:/opt/percona/bin
```

The --preferred-path 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>Config File Options</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>--prefetch-enabled</td>
<td>prefetch-enabled</td>
<td>Should the replicator service be setup as a prefetch applier</td>
<td>string</td>
</tr>
</tbody>
</table>

**--prefetch-max-time-ahead**

<table>
<thead>
<tr>
<th>Option</th>
<th>Config File Options</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>--prefetch-max-time-ahead</td>
<td>prefetch-max-time-ahead</td>
<td>Maximum number of seconds that the prefetch applier can get in front of the standard applier</td>
<td>numeric</td>
</tr>
</tbody>
</table>

**--prefetch-min-time-ahead**

<table>
<thead>
<tr>
<th>Option</th>
<th>Config File Options</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>--prefetch-min-time-ahead</td>
<td>prefetch-min-time-ahead</td>
<td>Minimum number of seconds that the prefetch applier must be in front of the standard applier</td>
<td>numeric</td>
</tr>
</tbody>
</table>

**--prefetch-schema**

<table>
<thead>
<tr>
<th>Option</th>
<th>Config File Options</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>--prefetch-schema</td>
<td>prefetch-schema</td>
<td>Schema to watch for timing prefetch progress</td>
<td>string</td>
</tr>
<tr>
<td>Default: tungsten_</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**--prefetch-sleep-time**

<table>
<thead>
<tr>
<th>Option</th>
<th>Config File Options</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>--prefetch-sleep-time</td>
<td>prefetch-sleep-time</td>
<td>How long to wait when the prefetch applier gets too far ahead</td>
<td>string</td>
</tr>
</tbody>
</table>

**--preview**

<table>
<thead>
<tr>
<th>Option</th>
<th>Config File Options</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>--preview</td>
<td>p</td>
<td>Displays the help message and preview the effect of the command line options</td>
<td>string</td>
</tr>
</tbody>
</table>
Command-line Tools

--profile file

<table>
<thead>
<tr>
<th>Option</th>
<th>--profile file [244]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>profile file [244]</td>
</tr>
<tr>
<td>Description</td>
<td>Sets name of config file (default: tungsten.cfg)</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

--profile-script

<table>
<thead>
<tr>
<th>Option</th>
<th>--profile-script [244]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>profile-script [244]</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>

--property

<table>
<thead>
<tr>
<th>Option</th>
<th>--property [244]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--property=key+=value [244], --property=key=value [244], --property=key~=/match/replace/ [244]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>property [244], property=key+=value [244], property=key=value [244], property=key~=/match/replace/ [244]</td>
</tr>
<tr>
<td>Description</td>
<td>Modify the value for key in any file that the configure script touches; key=value - Set key to value without evaluating template values or other rules; key+=value - Evaluate template values and then append value to the end of the line; key~=/match/replace/ - Evaluate template values then execute the specified Ruby regex with sub. For example --property=repliicator.key~=/(.*)/somevalue,\1/ will prepend 'somevalue' before the template value for 'repliicator.key'</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 [244]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>protect-configuration-files [244]</td>
</tr>
<tr>
<td>Description</td>
<td>When enabled, configuration files are protected to be only readable and updateable by the configured user</td>
</tr>
<tr>
<td>Value Type</td>
<td>boolean</td>
</tr>
<tr>
<td>Default</td>
<td>true</td>
</tr>
<tr>
<td>Valid Values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>false</td>
</tr>
<tr>
<td></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:

```
sh> ls -al /opt/continuent/tungsten/tungsten-replicator/conf/
total 148
drwxr-xr-x 2 tungsten mysql 4096 May 14 14:32 ..
drwxr-xr-x 11 tungsten mysql 4096 May 14 14:32 ...
drwxr-xr-x 1 tungsten mysql 33 May 14 14:32 dynamic-alpha.role
-rw-r--r-- 1 tungsten mysql 33 May 14 14:32 dynamic-alpha.role
-rw-r--r-- 1 tungsten mysql 5059 May 14 14:32 log4j.properties
-rw-r--r-- 1 tungsten mysql 3488 May 14 14:32 log4j-thl.properties
-rw-r--r-- 1 tungsten mysql 972 May 14 14:32 mysql-java-charsets.properties
-rw-r--r-- 1 tungsten mysql 420 May 14 14:32 replicator.service.properties
-rw-r----- 1 tungsten mysql 1590 May 14 14:35 services.properties
-rw-r----- 1 tungsten mysql 1590 May 14 14:35 .services.properties.orig
-rw-r----- 1 tungsten mysql 886 May 14 14:32 shard.list
-rw-r----- 1 tungsten mysql 4384 May 14 14:35 static-alpha.properties
-rw-r----- 1 tungsten mysql 4384 May 14 14:35 .static-alpha.properties.orig
-rw-r----- 1 tungsten mysql 5677 May 14 14:35 wrapper.conf
-rw-r----- 1 tungsten mysql 5677 May 14 14:35 .wrapper.conf.orig
```

When disabled, the files are readable by all users:

```
sh> ls -al /opt/continuent/tungsten/tungsten-replicator/conf/
total 148
drwxr-xr-x 2 tungsten mysql 4096 May 14 14:32 ..
drwxr-xr-x 11 tungsten mysql 4096 May 14 14:32 ...
```
### Command-line Tools

- **--quiet**
  - **Option:** `--quiet [245]`
  - **Aliases:** `--quiet [245]`
  - **Config File Options:** `q [245], quiet [245]`
  - **Description:** Only display warning and error messages
  - **Value Type:** string

- **--relay-directory**
  - **Option:** `--relay-directory [245]`
  - **Aliases:** `--repl-relay-directory [245]`
  - **Config File Options:** `relay-directory [245], repl-relay-directory [245]`
  - **Description:** Directory for logs transferred from the master
  - **Value Type:** string
  - **Default:** `{home directory}/relay`

- **--relay-enabled**
  - **Option:** `--relay-enabled [245]`
  - **Config File Options:** `relay-enabled [245]`
  - **Description:** Should the replicator service be setup as a relay master
  - **Value Type:** string

- **--relay-source**
  - **Option:** `--relay-source [245]`
  - **Aliases:** `--dataservice-relay-source [245]`
  - **Config File Options:** `dataservice-relay-source [245], relay-source [245]`
  - **Description:** Dataservice name to use as a relay source
  - **Value Type:** string

- **--remove-property=key**
  - **Option:** `--remove-property=key [245]`
  - **Config File Options:** `remove-property=key [245]`
  - **Description:** Remove a corresponding `--property` argument. Subcommands: defaults Modify the default values used for each data service or host
  - **Value Type:** string

- **--replication-password**
  - **Option:** `--replication-password [245]`
  - **Aliases:** `--datasource-password [245], --repl-datasource-password [245]`
  - **Config File Options:** `datasource-password [245], repl-datasource-password [245], replication-password [245]`
  - **Description:** Database password
<table>
<thead>
<tr>
<th>Value Type</th>
<th>string</th>
</tr>
</thead>
</table>

--replication-port

<table>
<thead>
<tr>
<th>Option</th>
<th>--replication-port [246]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--datasource-port [246], --repl-datasource-port [246]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>datasource-port [246], repl-datasource-port [246], replication-port [246]</td>
</tr>
<tr>
<td>Description</td>
<td>Database server port</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

--replication-user

<table>
<thead>
<tr>
<th>Option</th>
<th>--replication-user [246]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--datasource-user [246], --repl-datasource-user [246]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>datasource-user [246], repl-datasource-user [246], replication-user [246]</td>
</tr>
<tr>
<td>Description</td>
<td>Database login for Tungsten</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

--reset

<table>
<thead>
<tr>
<th>Option</th>
<th>--reset [246]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>reset [246]</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>

--rmi-port

<table>
<thead>
<tr>
<th>Option</th>
<th>--rmi-port [246]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-rmi-port [246]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>repl-rmi-port [246], rmi-port [246]</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 [246]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>rmi-user [246]</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 [246]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-role [246]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>repl-role [246], role [246]</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

<p>| Option             | --router-gateway-port [246] |</p>
<table>
<thead>
<tr>
<th>Config File Options</th>
<th>Description</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>router-gateway-port</td>
<td>The router gateway port</td>
<td>string</td>
</tr>
<tr>
<td>--router-jmx-port</td>
<td>--router-jmx-port</td>
<td>string</td>
</tr>
<tr>
<td>--security-directory</td>
<td>Storage directory for the Java security/encryption files</td>
<td>string</td>
</tr>
<tr>
<td>--service-alias</td>
<td>Replication alias of this dataservice</td>
<td>string</td>
</tr>
<tr>
<td>--service-type</td>
<td>What is the replication service type?</td>
<td>string</td>
</tr>
<tr>
<td>--skip-statemap</td>
<td>Do not copy the cluster-home/conf/statemap.properties from the previous install</td>
<td>string</td>
</tr>
<tr>
<td>--skip-validation-check</td>
<td>Do not run the specified validation check. Validation checks are identified by the string included in the error they output.</td>
<td>string</td>
</tr>
<tr>
<td>--skip-validation-warnings</td>
<td>Do not copy the cluster-home/conf/statemap.properties from the previous install</td>
<td>string</td>
</tr>
</tbody>
</table>
## Config File Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>skip-validation-warnings</code></td>
<td>Do not display warnings for the specified validation check. Validation checks are identified by the string included in the warning they output.</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

### `--slave-privileged-updates`

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--slave-privileged-updates</code></td>
<td>Does login for slave update have superuser privileges</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

### `--slaves`

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--slaves</code></td>
<td>What are the slaves for this dataservice?</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

### `--start`

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--start</code></td>
<td>Start the services after configuration</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

### `--start-and-report`

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--start-and-report</code></td>
<td>Start the services and report out the status after configuration</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

### `--svc-allow-any-remote-service`

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--svc-allow-any-remote-service</code></td>
<td>Replicate from any service</td>
<td>boolean</td>
<td>false</td>
</tr>
</tbody>
</table>

### `--svc-applier-buffer-size`

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--svc-applier-buffer-size</code></td>
<td>Applier block commit size (min 1)</td>
<td>numeric</td>
<td>10</td>
</tr>
</tbody>
</table>

<pagebreak>
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value Type</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>--svc-applier-filters</td>
<td>Replication service applier filters</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>--svc-extractor-filters</td>
<td>Replication service extractor filters</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>--svc-parallelization-type</td>
<td>Method for implementing parallel apply</td>
<td>string</td>
<td>disk, memory, none</td>
</tr>
<tr>
<td>--svc-shard-default-db</td>
<td>Mode for setting the shard ID from the default db</td>
<td>string</td>
<td>relaxed, stringent</td>
</tr>
<tr>
<td>--svc-table-engine</td>
<td>Replication service table engine</td>
<td>string</td>
<td>innodb</td>
</tr>
<tr>
<td>--svc-thl-filters</td>
<td></td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>Replication service THL filters</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**--target-dataservice**

<table>
<thead>
<tr>
<th>Option</th>
<th>--target-dataservice [250]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--slave-dataservice [250]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>slave-dataservice [250], target-dataservice [250]</td>
</tr>
<tr>
<td>Description</td>
<td>Dataservice to use to determine the value of host configuration</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

**--temp-directory**

<table>
<thead>
<tr>
<th>Option</th>
<th>--temp-directory [250]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>temp-directory [250]</td>
</tr>
<tr>
<td>Description</td>
<td>Temporary Directory</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

**--template-file-help**

<table>
<thead>
<tr>
<th>Option</th>
<th>--template-file-help [250]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config File Options</td>
<td>template-file-help [250]</td>
</tr>
<tr>
<td>Description</td>
<td>Display the keys that may be used in configuration template files</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

**--thl-directory**

<table>
<thead>
<tr>
<th>Option</th>
<th>--thl-directory [250]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-thl-directory [250]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>repl-thl-directory [250], thl-directory [250]</td>
</tr>
<tr>
<td>Description</td>
<td>Replicator log directory</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
<tr>
<td>Default</td>
<td>(home directory)/thl</td>
</tr>
</tbody>
</table>

**--thl-do-checksum**

<table>
<thead>
<tr>
<th>Option</th>
<th>--thl-do-checksum [250]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-thl-do-checksum [250]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>repl-thl-do-checksum [250], thl-do-checksum [250]</td>
</tr>
<tr>
<td>Description</td>
<td>Execute checksum operations on THL log files</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

**--thl-interface**

<table>
<thead>
<tr>
<th>Option</th>
<th>--thl-interface [250]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-thl-interface [250]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>repl-thl-interface [250], thl-interface [250]</td>
</tr>
<tr>
<td>Description</td>
<td>Listen interface to use for THL operations</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

**--thl-log-connection-timeout**

<table>
<thead>
<tr>
<th>Option</th>
<th>--thl-log-connection-timeout [250]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>--repl-thl-log-connection-timeout [250]</td>
</tr>
</tbody>
</table>
### Command-line Tools

#### Config File Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>repl-thl-log-connection-timeout</td>
<td>250, thl-log-connection-timeout [250]</td>
</tr>
<tr>
<td>Description</td>
<td>Number of seconds to wait for a connection to the THL log</td>
</tr>
<tr>
<td>Value Type</td>
<td>numeric</td>
</tr>
</tbody>
</table>

#### --thl-log-file-size

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>--thl-log-file-size</td>
<td>251</td>
</tr>
<tr>
<td>Aliases</td>
<td>repl-thl-log-file-size [251]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>repl-thl-log-file-size [251], thl-log-file-size [251]</td>
</tr>
<tr>
<td>Description</td>
<td>File size in bytes for THL disk logs</td>
</tr>
<tr>
<td>Value Type</td>
<td>numeric</td>
</tr>
</tbody>
</table>

#### --thl-log-fsync

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>--thl-log-fsync</td>
<td>251</td>
</tr>
<tr>
<td>Aliases</td>
<td>repl-thl-log-fsync [251]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>repl-thl-log-fsync [251], thl-log-fsync [251]</td>
</tr>
<tr>
<td>Description</td>
<td>Fsync THL records on commit. More reliable operation but adds latency to replication when using low-performance storage</td>
</tr>
<tr>
<td>Value Type</td>
<td>string</td>
</tr>
</tbody>
</table>

#### --thl-log-retention

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>--thl-log-retention</td>
<td>251</td>
</tr>
<tr>
<td>Aliases</td>
<td>repl-thl-log-retention [251]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>repl-thl-log-retention [251], thl-log-retention [251]</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>Default</td>
<td>7d</td>
</tr>
<tr>
<td>Valid Values</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>

#### --thl-protocol

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>--thl-protocol</td>
<td>251</td>
</tr>
<tr>
<td>Aliases</td>
<td>repl-thl-protocol [251]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>repl-thl-protocol [251], thl-protocol [251]</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>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>--topology</td>
<td>251</td>
</tr>
<tr>
<td>Aliases</td>
<td>dataservice-topology [251]</td>
</tr>
<tr>
<td>Config File Options</td>
<td>dataservice-topology [251], topology [251]</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>

#### --user

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>--user</td>
<td>251</td>
</tr>
</tbody>
</table>
## Troubleshooting

**ERROR >> node01 >> Unable to update the configuration of an installed directory**

When running `tpm update` command, it must be executed from a staging directory, not an installation directory.

### The `tungsten_provision_slave` Script

The script was added in Continuent Tungsten 2.0.1. 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 ] [ --force ] [ --help, -h ] [ --info, -i ] [ --json ] [ --net-ssh-option=key=value ] [ --notice, -n ] [ --offline ] [ --offline-timeout Integer ] [ --online ] [ --service String ] [ --source String ] [ --validate ] [ --verbose, -v ]
```

Where:

**Table 7.28. `tungsten_provision_slave` Command-line Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--clear-logs</code></td>
<td>Delete all THL and relay logs for the service</td>
</tr>
<tr>
<td><code>--direct</code></td>
<td>Use the MySQL data directory for staging and preparation</td>
</tr>
<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>--info, -i</code></td>
<td>Display info, notice, warning, and error messages</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 -f or --force options.

When provisioning masters, for example in fan-in in [Tungsten Replicator 2.2 Manual], multi-master, 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.

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.

7.11. The tungsten_monitor Script

The script was added in Continuent Tungsten 2.0.1. It cannot be backported to older versions.

The tungsten_monitor script provides a mechanism for monitoring the cluster state when monitoring tools like Nagios aren’t available. It implements the Tungsten Script Interface as well as these additional options.


Where:

Table 7.29. tungsten_monitor Command-line Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--check-log String</td>
<td>Email any lines in the log file that match the egrep expression. --check-log=tungsten-manager/log/tmsvc.log:OFFLINE</td>
</tr>
<tr>
<td>--connector-timeout String</td>
<td>Number of seconds to wait for a connector response</td>
</tr>
</tbody>
</table>
## Command-line Tools

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--dataservices String</td>
<td>This list of dataservices to monitoring to</td>
</tr>
<tr>
<td>--diagnostic-package</td>
<td>Create a diagnostic package if any issues are found</td>
</tr>
<tr>
<td>--directory</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>--disk String</td>
<td>Display a warning if any disk usage is above this percentage</td>
</tr>
<tr>
<td>--elb-script String</td>
<td>The xinetd script name that is responding to ELB liveness checks</td>
</tr>
<tr>
<td>--email String</td>
<td>Email address to send to when mailing any notifications</td>
</tr>
<tr>
<td>--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>--ignore String</td>
<td>Ignore notices that use this key</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>--latency String</td>
<td>The maximum allowed latency for replicators</td>
</tr>
<tr>
<td>--lock-dir String</td>
<td>Directory to store log and lock files in</td>
</tr>
<tr>
<td>--lock-timeout String</td>
<td>The number of minutes to sleep a notice after sending it</td>
</tr>
<tr>
<td>--mail String</td>
<td>Path to the mail program to use for sending messages</td>
</tr>
<tr>
<td>--max-backup-age String</td>
<td>Maximum age in seconds of valid backups</td>
</tr>
<tr>
<td>--net-ssh-option=key=value</td>
<td>Provide custom SSH options to use for communication to other hosts. A common example is --net-ssh-option=port=2222.</td>
</tr>
<tr>
<td>--notice, -n</td>
<td>Display notice, warning, and error messages</td>
</tr>
<tr>
<td>--reset</td>
<td>Remove all entries from the lock directory</td>
</tr>
<tr>
<td>--subject String</td>
<td>Email subject line</td>
</tr>
<tr>
<td>--validate</td>
<td>Only run script validation</td>
</tr>
<tr>
<td>--verbose, -v</td>
<td>Verbose</td>
</tr>
</tbody>
</table>

### General Operation

Each time the `tungsten_monitor` runs, it will run a standard set of checks. Additional checks may be turned on using command line options.

- Check that all Tungsten services for this host are running
- Check that all replication services and datasources are ONLINE
- Check that replication latency does not exceed a specified amount
- Check that the local connector is responsive
- Check disk usage

An example of adding it to crontab:

```shell
shell> crontab -e
10 * * * * /opt/continuent/tungsten/cluster-home/bin/tungsten_monitor >/dev/null 2>/dev/null
```

All messages will be sent to `/opt/continuent/share/tungsten_monitor/lastrun.log`.

### Sending results via email

The `tungsten_monitor` is able to send you an email when problems are found. It is suggested that you run the script as root so it is able to use the mail program without warnings.

Alerts are cached to prevent them from being sent multiple times and flooding your inbox. You may pass `--reset` to clear out the cache or `--lock-timeout` to adjust the amount of time this cache is kept. The default is 3 hours.

```shell
shell> crontab -e
10 * * * * /opt/continuent/tungsten/cluster-home/bin/tungsten_monitor --from=you@yourcompany.com \  --to=group@yourcompany.com >/dev/null 2>/dev/null
```

### Monitoring log files
The `tungsten_monitor` can optionally monitor log files for certain keywords. This example will alert you to any lines in `trepsvc.log` that include `OFFLINE`.

```
shell> tungsten_monitor --check-log=tungsten-replicator/log/trepsvc.log:OFFLINE
```

**Monitoring backup status**

Knowing you have a recent backup is an important part any Tungsten deployment. The `tungsten_monitor` will look for the latest backup across all datasources and compare it to the value `--max-backup-age`. This example will let you know if a valid backup has not been taken in 3 days.

```
shell> tungsten_monitor --max-backup-age=259200
```

**Compatibility**

The script only works with MySQL at this time.

### 7.12. The `tungsten_read_master_events` Script

The script was added in Continuent Tungsten 2.0.1. 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.

```
```

Where:

<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 String</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 String</code></td>
<td>Display events starting with this sequence number</td>
</tr>
<tr>
<td><code>--net-ssh-option=key=value</code></td>
<td>Provide custom SSH options to use for communication to other hosts. A common example is <code>--net-ssh-option=port=2222</code>.</td>
</tr>
<tr>
<td><code>--notice, -n</code></td>
<td>Display notice, warning, and error messages</td>
</tr>
<tr>
<td><code>--service String</code></td>
<td>Replication service to read information from</td>
</tr>
<tr>
<td><code>--source String</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
```

**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
```

**Compatibility**
The script only works with MySQL at this time.

The script was added in Continuent Tungsten 2.0.1 and Tungsten Replicator 2.2.0. It cannot be backported to older versions.

7.13. The **tungsten_set_position** Script

The script was added in Continuent Tungsten 2.0.1. 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.

```bash
```

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>--epoch String</td>
<td>The epoch number to use for updating the <code>trep_commit_seqno</code> table</td>
</tr>
<tr>
<td>--event-id String</td>
<td>The event id to use for updating the <code>trep_commit_seqno</code> table</td>
</tr>
<tr>
<td>--high String</td>
<td>Display events ending with this sequence number</td>
</tr>
<tr>
<td>--low String</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 Integer</td>
<td>Put required replication services offline after successful processing</td>
</tr>
<tr>
<td>--online</td>
<td>Put required replication services online after successful processing</td>
</tr>
<tr>
<td>--seqno String</td>
<td>The sequence number to use for updating the <code>trep_commit_seqno</code> table</td>
</tr>
<tr>
<td>--source String</td>
<td>Replication service to read information from</td>
</tr>
<tr>
<td>--source String</td>
<td>Determine metadata for the --after, --low, --high statements from this host</td>
</tr>
<tr>
<td>--source-id String</td>
<td>The source id to use for updating the <code>trep_commit_seqno</code> 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.

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.

```bash
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.

```bash
shell> tungsten_set_position --seqno=5273 --epoch=5264
                --source-id=db1
shell> tungsten_set_position --seqno=5273 --epoch=5264
```

256
Command-line Tools

---source-id=1234---event-123456-654321-0000000000000000

Compatibility

The script only works with MySQL at this time.

7.14. The tungsten_health_check Script

The script was added in Continuent Tungsten 2.0.1. It cannot be backported to older versions.

The **tungsten_health_check** may be used less frequently than Section 7.14, "The tungsten_health_check Script" to check the cluster against known best practices. It implements the Tungsten Script Interface as well as these additional options.

```
```

Where:

Table 7.32. **tungsten_health_check** Command-line Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--dataservices String</td>
<td>This list of dataservices to monitoring to</td>
</tr>
<tr>
<td>--diagnostic-package</td>
<td>Create a diagnostic package if any issues are found</td>
</tr>
<tr>
<td>--directory</td>
<td>The SCONTINUENT_ROOT directory to use for running this command. It will default to the directory you use to run the script.</td>
</tr>
<tr>
<td>--email String</td>
<td>Email address to send to when mailing any notifications</td>
</tr>
<tr>
<td>--force</td>
<td>Continue operation even if script validation fails</td>
</tr>
<tr>
<td>--from String</td>
<td>The from address for sending messages</td>
</tr>
<tr>
<td>--help, -h</td>
<td>Show help text</td>
</tr>
<tr>
<td>--ignore String</td>
<td>Ignore notices that use this key</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>--lock-dir String</td>
<td>Directory to store log and lock files in</td>
</tr>
<tr>
<td>--lock-timeout String</td>
<td>The number of minutes to sleep a notice after sending it</td>
</tr>
<tr>
<td>--mail String</td>
<td>Path to the mail program to use for sending messages</td>
</tr>
<tr>
<td>--net-ssh-option=key=value</td>
<td>Provide custom SSH options to use for communication to other hosts. A common example is --net-ssh-option=port=2222.</td>
</tr>
<tr>
<td>--notice, -n</td>
<td>Display notice, warning, and error messages</td>
</tr>
<tr>
<td>--show-differences</td>
<td>Show any differences in Tungsten configuration</td>
</tr>
<tr>
<td>--subject String</td>
<td>Email subject line</td>
</tr>
<tr>
<td>--test-failover</td>
<td>Test failover for each managed dataservice</td>
</tr>
<tr>
<td>--test-recover</td>
<td>Test recover for each managed dataservice</td>
</tr>
<tr>
<td>--test-switch</td>
<td>Test the switch command for each managed dataservice</td>
</tr>
<tr>
<td>--validate</td>
<td>Only run script validation</td>
</tr>
<tr>
<td>--verbose, -v</td>
<td>Verbose</td>
</tr>
</tbody>
</table>

Each time the **tungsten_health_check** runs, it will run a standard set of checks. Additional checks may be turned on using command line options.

- Check for errors using **tpm validate**
- Check that all servers in the dataservice are running the same version of Continuent Tungsten

The script can be run manually:

```
shell> tungsten_health_check
```
All messages will be sent to /opt/continuent/share/tungsten_health_check/lastrun.log.

Sending results via email

The `tungsten_health_check` is able to send you an email when problems are found. It is suggested that you run the script as root so it is able to use the mail program without warnings.

Alerts are cached to prevent them from being sent multiple times and flooding your inbox. You may pass `--reset` to clear out the cache or `--lock-timeout` to adjust the amount of time this cache is kept. The default is 3 hours.

```
shell> tungsten_health_check --from=you@yourcompany.com --to=group@yourcompany.com
```

Showing manual configuration file changes

The `tpm validate` command will fail if you have manually changed a configuration file. The file differences may be added if you include the `--show-differences` argument.

Testing Continuent Tungsten functionality

Continuent Tungsten includes a testing infrastructure that you can use at any time. By adding the `--test-switch`, `--test-failover` or `--test-recover` arguments to the command, we will test these operations on each database server.

Caution

This will have an impact on dataservice availability. Limit this operation to maintenance windows or times when you can experience managed outages.

Compatibility

The script only works with MySQL at this time.
Chapter 8. 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 8.1, "Filters: Pipeline Stages on Masters".

Figure 8.1. 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 8.2, "Filters: Pipeline Stages on Slaves".
Figure 8.2. 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:

```plaintext
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.

### 8.1. 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**
  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**
  Apply the filter between the internal queue and when the transactions are written to the THL. (*q-to-thl*).

- **--repl-svc-applier-filters**
  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 table `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=replicate \\
  --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 `stages` parameter to `trepctl`:

```
shell> trepctl status-name stages
Processing status command (stages)...
...
NAME       VALUE
----       -----       
applier.class : com.continuent.tungsten.replicator.applier.MySQLDrizzleApplier
applier.name : dbms
blockCommitRowCount: 10
committedDtnSeqno : 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
processedDtnSeqno : -1
procState : 0
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.

### 8.2. Enabling Additional Filters

The Continuent Tungsten 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 `RenameFilter` configuration is configured as follows:

```
replicator.filter.rename=com.continuent.tungsten.replicator.filter.RenameFilter
replicator.filter.rename.definitionsFile=${replicator.home.dir}/samples/extensions/java/rename.csv
```
The first line creates a new filter configuration using the corresponding Java class. In this case, the filter is named `rename`, as defined by the string `replicator.filter.rename`.

Configuration parameters for the filter are defined as values after the filter name. In this example, `definitionsFile` is the name of the property examined by the class to set the CSV file where the rename definitions are located.

To create an entirely new filter based on an existing filter class, a new property should be created with the new filter definition. Additional properties from this base should then be used. For example, to create a second `rename` filter definition called `custom`:

```
shell> /tools/tpm configure \
   --property='replicator.filter.rename.custom=com.continuent.tungsten.replicator.filter.RenameFilter' \
   --property='replicator.filter.rename.custom.definitionsFile=\n   /replicator_home_dir/samples/extensions/java/renamecustom.csv'
```

The filter can be enabled against the desired stage using the filter name `custom`:

```
shell> /tools/tpm configure \
   --repl-svc-applier-filters=custom
```

### 8.3. Filter Status

To determine which filters are currently being applied within a replicator, use the `trepctl status -name stages` command. This outputs a list of the current stages and their configuration. For example:

```
shell> trepctl status -name stages
Processing status command (stages)...
NAME                 VALUE
----                 -----  
applier.class      : com.continuent.tungsten.replicator.thl.THLStoreApplier
applier.name       : thl-applier
blockCommitRowCount: 1
committedMinSeqno  : 15
extractor.class    : com.continuent.tungsten.replicator.thl.RemoteTHLExtractor
extractor.name     : thl-remote
name               : remote-to-thl
processedMinSeqno  : -1
TaskCount          : 1
NAME                 VALUE
----                 -----  
applier.class      : com.continuent.tungsten.replicator.thl.THLParallelQueueApplier
applier.name       : parallel-q-applier
blockCommitRowCount: 10
committedMinSeqno  : 15
extractor.class    : com.continuent.tungsten.replicator.thl.THLStoreExtractor
extractor.name     : thl-extractor
name               : thl-to-q
processedMinSeqno  : -1
TaskCount          : 1
NAME                 VALUE
----                 -----  
applier.class      : com.continuent.tungsten.replicator.applier.MySQLDrizzleApplier
applier.name       : dbms
blockCommitRowCount: 10
committedMinSeqno  : 15
extractor.class    : com.continuent.tungsten.replicator.thl.THLParallelQueueExtractor
extractor.name     : parallel-q-extractor
filter.0.class     : com.continuent.tungsten.replicator.filter.TimeDelayFilter
filter.0.name      : delay
filter.1.class     : com.continuent.tungsten.replicator.filter.MySQLSessionSupportFilter
filter.1.name      : mysqlsessions
filter.2.class     : com.continuent.tungsten.replicator.filter.PrimaryKeyFilter
filter.2.name      : pkey
name               : q-to-dbms
processedMinSeqno  : -1
TaskCount          : 5
Finished status command (stages)...
```

In the output, the filters applied to the applier stage are shown in the last block of output. Filters are listed in the order in which they appear within the configuration.

For information about the filter operation and any modifications or changes made, check the `trepsvc.log` log file.

### 8.4. 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.
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 Continuent Tungsten is installed through tpm.

For the purposes of classification, the different filters have been identified 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>
<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>
</tbody>
</table>
### Replication Filters

<table>
<thead>
<tr>
<th>Filter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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> statements 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.

#### 8.4.1. BidiRemoteSlave 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>tpm</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>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.

#### 8.4.2. 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>tpm</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Row events only</td>
</tr>
</tbody>
</table>
### 8.4.3. BuildIndexTable Filter

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>buildindextable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.filter.BuildIndexTable</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.buildindextable</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td></td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Row events only</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>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>

### 8.4.4. CaseMapping Filter

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>casetransform</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.filter.CaseMappingFilter</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.casetransform</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td></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>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>

### 8.4.5. CDCMetadata Filter

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>customcdc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.filter.CDCMetadataFilter</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.customcdc</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td></td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Row events only</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>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>
</tbody>
</table>
### 8.4.6. 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>tpm Option compatibility</td>
<td>--svc-extractor-filters [249]</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Row events only</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>${replicator.global.extract.db.user}</td>
<td>The username for the connection to the database for looking up column definitions</td>
</tr>
<tr>
<td>password</td>
<td>string</td>
<td>${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}?createDB=true</td>
<td>JDBC URL of the database connection to use for looking up column definitions</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.000012:00000000000000004369:0
- SOURCEID = host31
- METADATA = [mysql_server_id=1;dbms_type=mysql;service=alpha;shard=test]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1]
- SQL(0) =
  - ACTION = INSERT
  - SCHEMA = test
  - TABLE = sales
  - ROW# = 0
  - COL(1) = 1
  - COL(2) = 23
  - COL(3) = 45
  - COL(4) = 45000.00
```

To a version where the column names are included as part of the THL record:

```
SEQ# = 43 / FRAG# = 0 (last frag)
```

---

**tableNameSuffix**

<table>
<thead>
<tr>
<th>string</th>
</tr>
</thead>
<tbody>
<tr>
<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>
</tbody>
</table>

**toSingleSchema**

<table>
<thead>
<tr>
<th>string</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creates and writes CDC data within a single schema</td>
</tr>
</tbody>
</table>

**sequenceBeginning**

<table>
<thead>
<tr>
<th>numeric</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sets the sequence number of the CDC data. The sequence is used to identify individual changesets in the CDC</td>
<td></td>
</tr>
</tbody>
</table>
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.

### 8.4.7. 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>tp</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Any event</td>
</tr>
<tr>
<td>Parameters</td>
<td>None</td>
</tr>
</tbody>
</table>

### 8.4.8. DatabaseTransform 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>tp</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>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>
### 8.4.9. 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.dumm</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></td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
</tr>
<tr>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

#### 8.4.10. 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>tpm Option compatibility</td>
<td>--repl-svc-extractor-filters [249]</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Row events only</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>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}?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;
<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Null</th>
<th>Key</th>
<th>Default</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>int(11)</td>
<td>NO</td>
<td>PRI</td>
<td>NULL</td>
<td>auto_increment</td>
</tr>
<tr>
<td>country</td>
<td>enum('US','UK','France','Australia')</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>city</td>
<td>int(11)</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>salesman</td>
<td>set('Alan','Zachary')</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>decimal(10,2)</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
</tbody>
</table>
```

When extracted in the THL, the representation uses the internal value (for example, 1 for the first enumerated value). This can be seen in the THL output below:

```
SEQ# = 138 / FRAG# = 0 (last frag)
- TIME = 2013-08-01 19:09:35.0
- EPOCH# = 122
- EVENTID = mysql-bin.000012:000000000021434:0
```
Replication Filters

For the `country` column, the corresponding value in the THL is `1`. With the `EnumToString` filter enabled, the value is expanded to the corresponding string value:

```
SEQ# = 121 / FRAG# = 0 (last frag)
TIME = 2013-08-01 19:05:14.0
EPOCH# = 102
EVENTID = mysql-bin.000012:0000000000018866:0
SOURCEID = host31
METADATA = [mysql_server_id=1;dbms_type=mysql;service=alpha;shard=test]
TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
OPTIONS = [foreign_key_checks = 1, unique_checks = 1]
SQL(0) =
  ACTION = INSERT
  SCHEMA = test
  TABLE = salesadv
  ROW# = 0
  COL(1: id) = 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 8.4.22, “SetToString Filter” and Section 8.4.6, “ColumnName Filter” filters.

### 8.4.11. 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>tpm</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Row events only</td>
</tr>
<tr>
<td>Parameters</td>
<td>None</td>
</tr>
</tbody>
</table>

### 8.4.12. 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>tpm</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Any event</td>
</tr>
<tr>
<td>Parameters</td>
<td>Parameter</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

269
**Replication Filters**

| **heartbeatInterval** | Numeric | 3000 | Interval in milliseconds when a heartbeat event is inserted into the THL |

### 8.4.13. 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><strong>tpm</strong> 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>

### 8.4.14. MySQLSessionSupport 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><strong>tpm</strong> 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>

### 8.4.15. 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><strong>tpm</strong> Option compatibility</td>
<td>--svc-extractor-filters [249], --svc-thl-filters [249], --svc-applier-filters [249]</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Row events only</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/networkclient.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, and creating the JSON configuration file.

### 8.4.15.1. 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"
         ]
      }
   ]
}
```
8.4.15.2. 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:

  ```json
  {
    "schema" : "hexdb",
    "transformation" : "String_to_HEX_v1",
    "service" : "firstrep",
    "type" : "filter",
    "payload" : 22,
    "row" : 0,
    "column" : "hexcol",
    "table" : "nextable",
    "seqno" : 145196,
    "fragments" : 1,
    "protocol" : "v0_9",
    "fragment" : 1
  }
  ``

  Where:

  - **protocol**
    The protocol version.
  - **service**
    The service name the requested the filter.
  - **type** [360]
    The message type, in this case, filter.
  - **row**
    The row of the source information from the THL that is being filtered.
  - **schema** [361]
    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** [359]
    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,
  "seqno": 145198,
  "table": "hextable",
  "service": "firstrep",
  "protocol": "v0.5",
  "schema": "hexdb",
  "payload": 8,
  "column": "hexcol",
  "row": 0
}
```

8.4.15.3. 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 $!" unless $socket;
  print "********
Server waiting for client connection on port 3112
******


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 \n\nReceived: <$data>\n*;

  # Decode the JSON part
  my $msg = decode_json($data);

  # Extract payload
  my $payload = undef;
  if ($msg->{payload} > 0)
  {
    print STDERR "$msg->{payload} bytes\n";
    $client_socket->read($payload, $msg->{payload});
    print "Payload: <$payload>\n";
  }
```

274
8.4.16. OptimizeUpdates Filter

The `optimizedupdates` 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><code>optimizedupdates</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td><code>com.continuent.tungsten.replicator.filter.OptimizeUpdatesFilter</code></td>
</tr>
<tr>
<td>Property prefix</td>
<td><code>replicator.filter.optimizeupdates</code></td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>Row events only</td>
</tr>
</tbody>
</table>

Parameters:

- `optimization` (optional)
The filter operates by removing column values for keys in the update statement that do not change. For example, when replicating the row event from the statement:

```
mysql> update testopt set msg = 'String1', string = 'String3' where id = 1;
```

Generates the following THL event data:

```
- SQL(0) =
- ACTION = UPDATE
- SCHEMA = test
- TABLE = testopt
- ROW# = 0
- COL(1: id) = 1
- COL(2: msg) = String1
- COL(3: string) = String3
- KEY(1: id) = 1
```

Column 1 (id) in this case is automatically implied by the KEY entry required for the update.

With the `optimizedupdates` filter enabled, the data in the THL is simplified to:

```
- SQL(0) =
- ACTION = UPDATE
- SCHEMA = test
- TABLE = testopt
- ROW# = 0
- COL(2: msg) = String1
- COL(3: string) = String4
- KEY(1: id) = 1
```

In tables where there are multiple keys the stored THL information can be reduced further.

**Warning**

The filter works by comparing the value of each KEY and COL entry in the THL and determining whether the value has changed or not. If the number of keys and columns do not match then the filter will fail with the following error message:

_Caused by: java.lang.Exception: Column and key count is different in this event! Cannot filter_

This may be due to a filter earlier within the filter configuration that has optimized or simplified the data. For example, the `pkey` filter removes KEY entries from the THL that are not primary keys, or `dropcolumns.js` which drops column data.

### 8.4.17. 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.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>pkey</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.filter.PrimaryKeyFilter</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.pkey</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>binlog-to-q</td>
</tr>
<tr>
<td>tpml Option compatibility</td>
<td>--repl-svc-extractor-filters[249]</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Row events only</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}:</td>
<td>JDBC URL of the database connection to use for looking up column definitions</td>
</tr>
</tbody>
</table>
### Replication Filters

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>addKeyToInsert</td>
<td>boolean</td>
<td>false</td>
</tr>
<tr>
<td>addColumnsToDeletes</td>
<td>boolean</td>
<td>false</td>
</tr>
</tbody>
</table>

- **addKeyToInsert**: Boolean. If set to true, primary keys are added to `INSERT` operations. This setting is required for batch loading.
- **addColumnsToDeletes**: Boolean. If set to true, full column metadata is added to `DELETE` operations. This setting is required for batch loading.

**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.

```plaintext
SEQ# = 142 / FRAG# = 0 (last frag)
- TIME = 2013-08-01 19:31:04.0
- EPOCH# = 122
- EVENTID = mysql-bin.000012:00000000000022187;0
- SOURCEID = host31
- METADATA = [mysql_server_id=1;dbms_type=mysql;service=alpha;shard=test]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1]
- SQL(0) =
  - ACTION = UPDATE
  - SCHEMA = test
  - TABLE = salesadv
  - ROW# = 0
  - COL(1: id) = 2
  - COL(2: country) = 1
  - COL(3: city) = 8374
  - COL(4: salesman) = 1
  - COL(5: value) = 89000.00
  - KEY(1: id) = 2
  - KEY(2: country) = 1
  - KEY(3: city) = 8374
  - KEY(4: salesman) = 1
  - KEY(5: value) = 89000.00
```

When the `PrimaryKey` is enabled, the key information has been optimized to only contain the actual primary keys are added to the row-based THL record:

```plaintext
SEQ# = 142 / FRAG# = 0 (last frag)
- TIME = 2013-08-01 19:31:04.0
- EPOCH# = 122
- EVENTID = mysql-bin.000012:00000000000022187;0
- SOURCEID = host31
- METADATA = [mysql_server_id=1;dbms_type=mysql;service=alpha;shard=test]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1]
- SQL(0) =
  - ACTION = UPDATE
  - SCHEMA = test
  - TABLE = salesadv
  - ROW# = 0
  - COL(1: id) = 2
  - COL(2: country) = 1
  - COL(3: city) = 8374
  - COL(4: salesman) = 1
  - COL(5: value) = 89000.00
  - KEY(1: id) = 2
```

The final line shows the addition of the primary key `id` added to THL event.

The two options, `addKeyToInsert` 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.

### 8.4.18. PrintEvent Filter

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-configured filter name</td>
<td>printevent</td>
</tr>
<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></td>
</tr>
</tbody>
</table>
### 8.4.19. Rename Filter

The RenameFilter 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.

<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>
</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
./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.

For example, the specification:

```
*,chicago,*,-,newyork,-
```

Would rename the table `chicago` in every database schema to `newyork`. The schema and column names are not modified.

The specification:

```
*,chicago,destination, -, -, source
```

Would match all schemas, but update the column `destination` in the table `chicago` to the column name `source`, without changing the schema or table name.
Processing of the individual rules is executed in a specific order to allow for complex matching and application of the rename changes.

- Rules are case sensitive.
- Schema names are looked up in the following order:
  1. `schema.table` (explicit schema/table)
  2. `schema.*` (explicit schema, wildcard table)
- Table names are looked up in the following order:
  1. `schema.table` (explicit schema/table)
  2. `*.table` (wildcard schema, explicit table)
- Column names are looked up in the following order:
  1. `schema.table` (explicit schema/table)
  2. `schema.*` (explicit schema, wildcard table)
  3. `*.table` (wildcard schema, explicit table)
  4. `*.*` (wildcard schema, wildcard table)
- Rename operations match the first specification according to the above rules, and only one matching rule is executed.

### 8.4.19.1. 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`. 
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
```

### 8.4.20. ReplicateColumns Filter

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>replicatocolumns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classname</td>
<td>com.continuent.tungsten.replicator.filter.ReplicateColumnsFilter</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.replicatocolumns</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>Row events only</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>ignore</code></td>
<td>string</td>
<td>empty</td>
<td>Comma separated list of tables and optional column names to ignore during replication</td>
</tr>
<tr>
<td><code>do</code></td>
<td>string</td>
<td>empty</td>
<td>Comma separated list of tables and optional column names to replicate</td>
</tr>
</tbody>
</table>

### 8.4.21. 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>com.continuent.tungsten.replicator.filter.ReplicateFilter</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.replicate</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>Any</td>
</tr>
<tr>
<td><code>tpm</code> Option compatibility</td>
<td></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>ignore</code></td>
<td>string</td>
<td>empty</td>
<td>Comma separated list of database/tables to ignore during replication</td>
</tr>
</tbody>
</table>
Replication Filters

<table>
<thead>
<tr>
<th>do</th>
<th>string</th>
<th>empty</th>
<th>Comma separated list of database/tables to replicate</th>
</tr>
</thead>
</table>

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 honoured. 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.

8.4.22. SetToString Filter

The `SetToString` filter 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>
</table>

Classname: `com.continuent.tungsten.replicator.filter.SetToStringFilter`

Property prefix: `replicator.filter.settoString`

Stage compatibility: `binlog-to-tpm`

tpm Option compatibility: `--repl-svc-extractor-filters`

Data compatibility: Row events only

<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}?createDB=true</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 

\textbf{SET} column, \textit{salesman}:

\begin{verbatim}
mysql> describe salesadv;
+----------+--------------------------------------+------+-----+---------+----------------+
| Field    | Type                                 | Null | Key | Default | Extra          |
|----------+--------------------------------------+------+-----+---------+----------------+
| id       | int(11)                              | NO   | PRI | NULL    | auto_increment |
| country  | enum('US','UK','France','Australia') | YES  |     | NULL    |                |
| city     | int(11)                              | YES  |     | NULL    |                |
| salesman | set('Alan','Zachary')                | YES  |     | NULL    |                |
| value    | decimal(10,2)                        | YES  |     | NULL    |                |
|----------+--------------------------------------+------+-----+---------+----------------+
\end{verbatim}

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.

\begin{verbatim}
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
\end{verbatim}

For the \textit{salesman} column, the corresponding value in the THL is \textit{1}. With the \textit{SetToString} filter enabled, the value is expanded to the corresponding string value:

\begin{verbatim}
SEQ# = 121 / FRAG# = 0 (last frag)
- TIME = 2013-08-01 19:05:14.0
- EPOCH# = 102
- EVENTID = mysql-bin.000012:0000000000018866:0
- SOURCEID = host31
- METADATA = [mysql_server_id=1;dbms_type=mysql;service=alpha;shard=test]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
- OPTIONS = [foreign_key_checks = 1, unique_checks = 1]
- SQL(0) = 
- ACTION = INSERT
- SCHEMA = test
- TABLE = salesadv
- ROW# = 0
  - COL(1: id) = 1
  - COL(2: country) = US
  - COL(3: city) = 8374
  - COL(4: salesman) = Alan
  - COL(5: value) = 35000.00
\end{verbatim}

The examples here also show the Section 8.4.10, “EnumToString Filter” and Section 8.4.6, “ColumnName Filter” filters.

### 8.4.23. 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>
<tr>
<td>Stage compatibility</td>
<td></td>
</tr>
<tr>
<td>tpm Option compatibility</td>
<td>Any event</td>
</tr>
<tr>
<td>Data compatibility</td>
<td></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>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>
</tbody>
</table>
8.4.24. TimeDelay 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>
<tr>
<td>Data compatibility</td>
<td>Any event</td>
</tr>
<tr>
<td>Parameters</td>
<td>delay</td>
</tr>
<tr>
<td>Type</td>
<td>numeric</td>
</tr>
<tr>
<td>Default</td>
<td>300</td>
</tr>
<tr>
<td>Description</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.

8.5. 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**

Updating the static properties file for the replicator will break automated upgrades through `tpm`. When upgrading, `tpm` relies on existing template files to create the new configuration based on the `tpm` parameters used.

Making a backup copy of the configuration file automatically generated by `tpm`, and then using this before performing an **upgrade** will enable you to update your configuration automatically. Settings for the JavaScript filter will then need to be updated in the configuration file manually.
To enable a JavaScript filter that has not already been configured, the static properties file (static-SERVICE.properties) must be edited to include the definition of the filter using the JavaScriptFilter class, using the script property to define the location of the actual JavaScript file containing the filter definition. For example, the supplied ansiquotes.js filter is defined as follows:

```
replicator.filter.ansiquotes=com.continuent.tungsten.replicator.filter.JavaScriptFilter
replicator.filter.ansiquotes.script=${replicator.home.dir}/samples/extensions/javascript/ansiquotes.js
```

To use the filter, add the filter name, ansiquotes in the above example, to the required stage:

```
replicator.stage.q-to-dbms.filters=mysqlsessions,pkey,bidiSlave,ansiquotes
```

Then restart the replicator to enable the configuration:

```
shell> replicator restart
```

Note
---

This procedure will need to be enabled on each replicator that you want to use the JavaScript filter.

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.

### 8.5.1. 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) {
      // 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.

### 8.5.1.1. 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.

### 8.5.1.2. 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 to 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 filter’s own context, use the `filterProperties` class with the `getString()` method. For example, the `dbrename.js` 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}/samples/extensions/javascript/dbrename.js
  replicator.filter.jsdbrename.dbsource=contacts
  replicator.filter.jsdbrename.dbtarget=nyc_contacts
  ```

  Within the JavaScript filter, they are retrieved using:

  ```
  sourceName = filterProperties.getString("dbsource");
  targetName = filterProperties.getString("dbtarget");
  ```

- **Generic Replicator Properties**
  
  General properties can be retrieved using the `properties` class and the `getString()` method:

  ```
  master = properties.getString("replicator.thl.remote_uri");
  ```

### 8.5.1.3. 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:
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.

### 8.5.1.4. 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:

```
+----------------+   +---------------+   +----------------+   +----------------+   +----------------+   +----------------+
| ReplDBMSEvent |   | DBMSData      |   | StatementData  |   | DBMSData        |   | RowChangeData  |   | OneRowChange   |
| DBMSData       |   | StatementData |   | DBMSData        |   | RowChangeData  |   | OneRowChange   |   | OneRowChange   |
| DBMSData       |   | RowChangeData |   | StatementData   |   | DBMSData        |   | RowChangeData  |   | OneRowChange   |
| ReplDBMSEvent  |   | StatementData |   | OneRowChange    |   | OneRowChange    |   | OneRowChange    |   | OneRowChange    |
```

A single event can contain both statement and row change information within the list of individual `DBMSData` events. An event or

### 8.5.1.4.1. `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><code>getAppliedLatency()</code></td>
<td>Returns the latency of the embedded event. See Section D.2.6, “Terminology: Fields appliedLatency”</td>
</tr>
<tr>
<td><code>getData()</code></td>
<td>Returns an array of the <code>DBMSData</code> Objects within the event</td>
</tr>
<tr>
<td><code>getDBMSEvent()</code></td>
<td>Returns the original <code>DBMSEvent</code> object</td>
</tr>
</tbody>
</table>
### Replication Filters

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getEpochNumber()</code></td>
<td>Get the Epoch number of the stored event. See THL EPOCH# [359]</td>
</tr>
<tr>
<td><code>getEventId()</code></td>
<td>Returns the native event ID. See THL EVENTID [360]</td>
</tr>
<tr>
<td><code>getExtractedTstamp()</code></td>
<td>Returns the timestamp of the event.</td>
</tr>
<tr>
<td><code>getFragno()</code></td>
<td>Returns the fragment ID. See THL SEQNO [359]</td>
</tr>
<tr>
<td><code>getLastFrag()</code></td>
<td>Returns true if the fragment is the last fragment in the event.</td>
</tr>
<tr>
<td><code>getSeqno()</code></td>
<td>Returns the native sequence number. See THL SEQNO [359]</td>
</tr>
<tr>
<td><code>getShardId()</code></td>
<td>Returns the shard ID for the event.</td>
</tr>
<tr>
<td><code>getSourceId()</code></td>
<td>Returns the source ID of the event. See THL SOURCEID [360]</td>
</tr>
<tr>
<td><code>setShardId()</code></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(i);
```

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.

### Updating the Shard ID

The `setShardId()` method can also be used to set the shard ID within an event. This can be used in filters where the shard ID is updated by examining the schema or table being updated within the embedded SQL or row data. An example of this is provided in Section 8.5.2.16, “shardbytable.js Filter”.

### 8.5.1.4.2. `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:

```javascript
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 8.5.1.4.3, "StatementData Objects". For row data, see Section 8.5.1.4.4, "RowChangeData Objects".

8.5.1.4.3. 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>getQuery()</td>
<td>Returns the SQL statement</td>
</tr>
<tr>
<td>setQuery()</td>
<td>Updates the SQL statement</td>
</tr>
<tr>
<td>appendToQuery()</td>
<td>Appends a string to an existing query</td>
</tr>
<tr>
<td>getDefaultSchema()</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>setDefaultSchema()</td>
<td>Set the default schema for the SQL statement</td>
</tr>
<tr>
<td>getTimestamp()</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 8.5.2.3, "dbrename.js Filter".

8.5.1.4.4. 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 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 object.</td>
</tr>
<tr>
<td>getRowChanges()</td>
<td>Returns an array list of all the changes as OneRowChange objects.</td>
</tr>
</tbody>
</table>
Replication Filters

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 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 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
rowchange.setSchemaName('prefix_' + rowchange.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:

```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()` method returns the column specification of the row change. The information is returned as an array of the individual columns and their specification:

```
columns = oneRowChange.getColumnSpec();
```

For each column specification a `ColumnSpec` object is returned, which supports the following methods:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getIndex()</td>
<td>Gets the index of the column within the row change</td>
</tr>
<tr>
<td>getLength()</td>
<td>Gets the length of the column</td>
</tr>
<tr>
<td>getName()</td>
<td>Returns the column name if available</td>
</tr>
<tr>
<td>getType()</td>
<td>Gets the type number of the column</td>
</tr>
<tr>
<td>getTypeDescription()</td>
<td></td>
</tr>
<tr>
<td>isBlob()</td>
<td>Returns true if the column is a blob</td>
</tr>
<tr>
<td>isNotNull()</td>
<td>Returns true if the column is configured as <code>NOT NULL</code></td>
</tr>
<tr>
<td>isUnsigned()</td>
<td>Returns true if the column is unsigned.</td>
</tr>
<tr>
<td>setBlob()</td>
<td>Set the column blob specification</td>
</tr>
<tr>
<td>setIndex()</td>
<td>Set the column index order</td>
</tr>
<tr>
<td>setLength()</td>
<td>Returns the column length</td>
</tr>
<tr>
<td>setName()</td>
<td>Set the column name</td>
</tr>
<tr>
<td>setNotNull()</td>
<td>Set whether the column is configured as <code>NOT NULL</code></td>
</tr>
<tr>
<td>setSigned()</td>
<td>Set whether the column data is signed</td>
</tr>
<tr>
<td>setType()</td>
<td>Set the column type</td>
</tr>
<tr>
<td>setTypeDescription()</td>
<td>Set the column type description</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</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>VARCHAR</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>TEXT</td>
<td>2004</td>
<td>Use <code>isBlob()</code> to identify if the column is a blob or not</td>
</tr>
<tr>
<td>TIME [360]</td>
<td>92</td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>91</td>
<td></td>
</tr>
<tr>
<td>DATETIME</td>
<td>92</td>
<td></td>
</tr>
<tr>
<td>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.

### Extracting Row Data

The `getColumnValues()` method returns an array that corresponds to the information returned by the `getColumnSpec()` method. That is, the method returns a complementary array of the row change values, one element for each row, where each row is itself a further array of each column:

```
values = oneRowChange.getColumnValues();
```

This means that index 0 of the array from `getColumnSpec()` refers to the same column as index 0 of the array for a single row from `getColumnValues()`.

<table>
<thead>
<tr>
<th>getColumnSpec()</th>
<th>msgid</th>
<th>message</th>
<th>msgdate</th>
</tr>
</thead>
<tbody>
<tr>
<td>getColumnValues()</td>
<td></td>
<td></td>
<td></td>
</tr>
<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.js sample, dates with a zero value are set to NULL using the following code:

```javascript
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.

### 8.5.2. JavaScript Filter Reference

Tungsten Replicator comes with a number of JavaScript filters that can either be used directly, or that can be modified and adapted to suit individual requirements. The majority of these scripts are located in `tungsten-replicator/samples/extensions/javascript`, more advanced scripts are located in `tungsten-replicator/samples/scripts/javascript-advanced`.

#### 8.5.2.1. ansiquotes.js Filter

The ansiquotes.js script 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/samples/extensions/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>tpm Option compatibility</td>
<td>--svc-extractor-filters [249]</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Any event</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
</tr>
</tbody>
</table>

This changes a statement such as:

```plaintext
INSERT INTO notepad VALUES ('message', 0);
```

To:
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='';"
newStatement = new com.continuent.tungsten.replicator.dbms.StatementData(
    query,
    null,
    null
);
data.add(data.size(), newStatement);
```

### 8.5.2.2. breadcrumbs.js Filter

The `breadcrumbs.js` filter records regular 'breadcrumb' points into a MySQL table for systems that do not have global transaction IDs. This can be useful if recovery needs to be made to a specific point. The example also shows how metadata information for a given event can be updated based on the information from a table.

**Pre-configured filter name**

ansiquotes

**JavaScript Filter File**

tungsten-replicator/samples/extensions/javascript/breadcrumbs.js

**Property prefix**

replicator.filter.ansiquotes

**Stage compatibility**

binlog-to-q

**tpm Option compatibility**

--svc-extractor-filters

**Data compatibility**

Any event

<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>(not specified)</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);
```

2. 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 work in two stages, and operates because the JavaScript instance is persistent as long as the Replicator is running. This means that data extracted during replication stays in memory and can be applied to later transactions. Hence the breadcrumb ID and offset information can be identified and used on each call to the filter function.

The first part of the filter event identifies the breadcrumb table and extracts the identified breadcrumb counter:
if (table.compareToIgnoreCase("breadcrumbs") == 0)
{
    columnValues = oneRowChange.getColumnValues();
    for (row = 0; row < columnValues.size(); row++)
    {
        values = columnValues.get(row);
        server_id_value = values.get(0);
        if (server_id == null || server_id == server_id_value.getValue())
        {
            counter_value = values.get(1);
            breadcrumb_counter = counter_value.getValue();
            breadcrumb_offset = 0;
        }
    }
}

The second part updates the event metadata using the extracted breadcrumb information:

topLevelEvent = event.getDBMSEvent();
if (topLevelEvent != null)
{
    xact_server_id = topLevelEvent.getMetadataOptionValue("mysql_server_id");
    if (server_id == xact_server_id)
    {
        topLevelEvent.setMetaDataOption("breadcrumb_counter", breadcrumb_counter);
        topLevelEvent.setMetaDataOption("breadcrumb_offset", breadcrumb_offset);
    }
}

To calculate the offset (i.e. the number of events since the last breadcrumb value was extracted), the script determines if the event was the last fragment processed, and updates the offset counter:

if (!event.getLastFrag())
{
    breadcrumb_offset = breadcrumb_offset + 1;
}

8.5.2.3. dbrename.js Filter

The dbrename.js JavaScript filter renames database (schemas) using two parameters from the properties file, the dbsource and dbtarget. Each event is then processed, and the statement or row based schema information is updated to dbtarget when the dbsource schema is identified.

| Pre-configured filter name | not configured |
| JavaScript Filter File | tungsten-replicator/samples/extensions/javascript/dbrename.js |
| Property prefix | |
| Stage compatibility | binlog-to-q |
| tpm Option compatibility | --svc-extractor-filters[249] |
| 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>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:

replicator.filter.dbrename=com.continuent.tungsten.replicator.filter.JavaScriptFilter
replicator.filter.dbrename.script=${replicator.home.dir}/samples/extensions/javascript/dbrename.js
replicator.filter.dbrename.dbsource=SOURCE
replicator.filter.dbrename.dbtarget=TEST

The operation of the filter is straightforward, because the schema name is exposed and settable within the statement and row change objects:

function filter(event)
{
    sourceName = filterProperties.getString("dbsource");
    targetName = filterProperties.getString("dbtarget");
    data = event.getData();
    for(i=0;i<data.size();i++)
Replication Filters

8.5.2.4. `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.js` 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>(not configured)</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript Filter File</td>
<td><code>tungsten-replicator/samples/extensions/javascript/dbselector.js</code></td>
</tr>
<tr>
<td>Property prefix</td>
<td></td>
</tr>
<tr>
<td>Stage compatibility</td>
<td><code>binlog-to-q, q-to-thl, q-to-dbms</code></td>
</tr>
<tr>
<td><code>tpm</code> Option compatibility</td>
<td><code>--svc-extractor-filters [249], --svc-applier-filters [249]</code></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><code>db</code></td>
<td>String</td>
</tr>
</tbody>
</table>

Within the filter, statement changes look for the schema in the `StatementData` object and remove it from the array:

```java
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);
        }
    }
}
```

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--;
        }
    }
}
```
8.5.2.5. dbupper.js Filter

The dbupper.js script 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>(not configured)</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript Filter File</td>
<td>tungsten-replicator/samples/extensions/javascript/dbupper.js</td>
</tr>
<tr>
<td>Property prefix</td>
<td></td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>binlog-to-q</td>
</tr>
<tr>
<td>tpm Option compatibility</td>
<td>--svc-extractor-filters[249], --svc-applier-filters[249]</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>definitionsFile</td>
<td>Filename</td>
<td>(none)</td>
<td>Database name to be converted to uppercase</td>
</tr>
</tbody>
</table>

For example, within statement data:

```javascript
from = d.getDefaultSchema();
if (from != null) {
    to   = from.toUpperCase();
    d.setDefaultSchema(to);
}
```

8.5.2.6. dropcolumn.js Filter

The dropcolumn.js 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/samples/extensions/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-dbsns</td>
</tr>
<tr>
<td>tpm Option compatibility</td>
<td>--svc-extractor-filters[249], --svc-applier-filters[249]</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>definitionsFile</td>
<td>Filename</td>
<td>~/dropcolumn.js</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
```

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:

```json
[
  {
    "schema": "vip",
    "table": "clients",
    "column": [
      "personal_code",
      "birth_date",
      "email"
    ]
  }
]
```
Replication Filters

Where:

• **schema** [361] 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:

```json
[
    {
        "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:

```json
[
    {
        "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`.

### 8.5.2.7. *dropcomments.js* Filter

The *dropcomments.js* script removes comments from statements within the event data.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>dropcomments</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript Filter File</td>
<td>tungsten-replicator/samples/extensions/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>tpm Option compatibility</td>
<td>--svc-extractor-filters [249], --svc-applier-filters [249]</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>

Row changes do not have comments, so the script only has to change the statement information, which is achieved by using a regular expression:

```javascript
sqlOriginal = d.getQuery();
sqlNew = sqlOriginal.replaceAll(/\*\*(?:.|
```
To handle the case where the statement could only be a comment, the statement is removed:

```java
if(sqlNew.trim().length()==0)
{
   data.remove(i);
   i--;
}
```

### 8.5.2.8. dropmetadata.js Filter

All events within the replication stream contain metadata about each event. This information can be individual processed and manipulated. The `dropmetadata.js` script removes specific metadata from each event, configured through the `option` parameter to the filter.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>option</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:

```java
metaData = event.getDBMSEvent().getMetadata();
for(m = 0; m < metaData.size(); m++)
{
   option = metaData.get(m);
   if(option.getOptionName().compareTo(optionName)==0)
   {
      metaData.remove(m);
      break;
   }
}
```

### 8.5.2.9. 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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>option</td>
<td>String</td>
<td>(none)</td>
<td>Name of the metadata field to be dropped</td>
</tr>
</tbody>
</table>

This is achieved by checking for statements, and then removing them from the event:

```java
data = event.getData();
for(i = 0; i < data.size(); i++)
{
   d = data.get(i);
   if(d instanceof com.continuent.tungsten.replicator.dbms.StatementData)
   {
      data.remove(i);
      i--;
   }
```
8.5.2.10. foreignkeychecks.js Filter

The foreignkeychecks.js script switches off foreign key checks for statements using the following statements:

- `CREATE TABLE`
- `DROP TABLE`
- `ALTER TABLE`
- `RENAME TABLE`

**Pre-configured filter name**: `foreignkeychecks`

**JavaScript Filter File**: `tungsten-replicator/samples/extensions/javascript/foreignkeychecks.js`

**Property prefix**: `replicator.filter.foreignkeychecks`

**Stage compatibility**: `binlog-to-q, q-to-dbms`

**tpm Option compatibility**: `--svc-extractor-filters [249], --svc-applier-filters [249]`

**Data compatibility**: Any event

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</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.

8.5.2.11. insertsonly.js Filter

The insertsonly.js script filters events to only include ROW-based events using `INSERT`.

**Pre-configured filter name**: (not configured)

**JavaScript Filter File**: `tungsten-replicator/samples/extensions/javascript/insertonly.js`

**Property prefix**: `replicator.filter.insertonly`

**Stage compatibility**: `q-to-dbms`

**tpm Option compatibility**: `--svc-applier-filters [249]`

**Data compatibility**: Row events only

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>

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--;
}
```
### 8.5.2.12. nocreatedbifnotexists.js Filter

The nocreatedbifnotexists.js script removes statements that start with:

```
CREATE DATABASE IF NOT EXISTS
```

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>(not configured)</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript Filter File</td>
<td><code>tungsten-replicator/samples/extensions/javascript/nocreatedbifnotexists.js</code></td>
</tr>
<tr>
<td>Property prefix</td>
<td></td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>q-to-dbms</td>
</tr>
<tr>
<td>tpm Option compatibility</td>
<td><code>--svc-applier-filters [249]</code></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>
</table>

This can be useful in heterogeneous replication where tungsten specific databases need to be removed from the replication stream.

The script works in two phases. The first phase creates a global variable within the `prepare()` function that defines the string to be examined:

```javascript
function prepare()
{
    beginning = "CREATE DATABASE IF NOT EXISTS";
}
```

Row based changes can be ignored, but for statement based events, the SQL is examine and the statement removed if the SQL starts with the text in the `beginning` variable:

```javascript
sql = d.getQuery();
if(sql.startsWith(beginning))
{
    data.remove(i);
    i--;
}
```

### 8.5.2.13. noonlykeywords.js Filter

The `ONLY` keyword is used within PostgreSQL to update only the specified table (and no sub-tables) within a given SQL statement. This is invalid SQL within MySQL. The noonlykeywords.js filter removes this keyword from statements and can be used in PostgreSQL to MySQL replication topologies.

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>(not configured)</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript Filter File</td>
<td><code>tungsten-replicator/samples/extensions/javascript/noonlykeywords.js</code></td>
</tr>
<tr>
<td>Property prefix</td>
<td></td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>q-to-dbms</td>
</tr>
<tr>
<td>tpm Option compatibility</td>
<td><code>--svc-applier-filters [249]</code></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>
</table>

The script operates by examining the statement data and then using a regular expression to remove the `ONLY` keyword. The updated query is then set to the updated SQL:

```javascript
sqlOriginal = d.getQuery();
if(sqlOriginal.toUpperCase().startsWith("DELETE FROM ONLY") ||
    sqlOriginal.toUpperCase().startsWith("UPDATE ONLY")
)
{
    sqlNew = sqlOriginal.replaceFirst(" (?i)ONLY", "")
    d.setQuery(sqlNew);
}
```

### 8.5.2.14. pgddl.js Filter

The pgddl.js filter updates SQL statements so that MySQL DDL statements are updated to a PostgreSQL compatible DDL statement.
Replication Filters

<table>
<thead>
<tr>
<th>Pre-configured filter name</th>
<th>(not configured)</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript Filter File</td>
<td>tungsten-replicator/samples/extensions/javascript/pgddl.js</td>
</tr>
<tr>
<td>Property prefix</td>
<td></td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>g-to-dbms</td>
</tr>
<tr>
<td>tpm Option compatibility</td>
<td>--svc-applier-filters[249]</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 script operates in two stages. The first is called within the `prepare()` function, creating a two-dimensional array containing the MySQL statement fragment and corresponding PostgreSQL fragment that should replace it.

```javascript
function prepare()
{
    transformers = new Array();
    transformers[0] = new Array(2);
    transformers[0][0] = " integer auto_increment ";
    transformers[0][1] = " serial ";
    ...
}
```

Within the statement processing, a replace function is called for each `transformers` element to replace the text, and then updates the SQL in the object:

```javascript
newSql = sql.replace(transformers[t][0], transformers[t][1]);
d.setQuery(newSql);
```

### 8.5.2.15. shardbyseqno.js Filter

Shards within the replicator enable data to be parallelised 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/samples/extensions/javascript/shardbyseqno.js</td>
</tr>
<tr>
<td>Property prefix</td>
<td>replicator.filter.shardbyseqno</td>
</tr>
<tr>
<td>Stage compatibility</td>
<td>g-to-dbms</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.js` 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:

```javascript
event.setShardId(event.getSeqno() % shards);
```

**Note**

Care should be taken with this script, 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.

### 8.5.2.16. 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.js` 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.
Replication Filters

### JavaScript Filter File

<table>
<thead>
<tr>
<th>Property prefix</th>
<th>replicator.filter.shardbytable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stage compatibility</td>
<td>q-to-dbms</td>
</tr>
<tr>
<td>tpm Option compatibility</td>
<td>--svc-applier-filters [249]</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>oneRowChange</td>
<td>Field</td>
<td></td>
<td></td>
</tr>
<tr>
<td>schemaName</td>
<td>String</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tableName</td>
<td>String</td>
<td></td>
<td></td>
</tr>
<tr>
<td>id</td>
<td>String</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The key part of the filter is the extraction and construction of the ID, which occurs during row processing:

```javascript
oneRowChange = rowChanges.get(j);
schemaName = oneRowChange.getSchemaName();
tableName = oneRowChange.getTableName();
id = schemaName + "_" + tableName;
if (proposedShardId == null)
{
    proposedShardId = id;
}
```

#### 8.5.2.17. tosilledb.js Filter

This filter updates the replicated information so that it goes to an explicit schema, as defined by the user. The filter can be used to combine multiple tables to a single schema.

| Pre-configured filter name | (not configured) |
| JavaScript Filter File | tungsten-replicator/samples/extensions/javascript/tosilledb.js |
| Property prefix | replicator.filter.ansiquotes |
| Stage compatibility | q-to-dbms |
| tpm Option compatibility | --svc-applier-filters [249] |
| Data compatibility | Any event |

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>db</td>
<td>String</td>
<td>(none)</td>
<td>Database name into which to replicate all tables</td>
</tr>
<tr>
<td>skip</td>
<td>String</td>
<td>(none)</td>
<td>Comma-separated list of databases to be ignored</td>
</tr>
</tbody>
</table>

A database can be optionally ignored through the `skip` parameter within the configuration:

```javascript
replicator.filter.tosilledb=com.continuent.tungsten.replicator.filter.JavaScriptFilter
replicator.filter.tosilledb=com.continuent.tungsten.replicator.filter.JavaScriptFilter
replicator.filter.tosilledb=com.continuent.tungsten.replicator.filter.JavaScriptFilter
replicator.filter.tosilledb=com.continuent.tungsten.replicator.filter.JavaScriptFilter
replicator.filter.tosilledb=com.continuent.tungsten.replicator.filter.JavaScriptFilter
```

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:

```javascript
if(oldDb!=null && oldDb.compareTo(skip)!=0)
{
    d.setDefaultSchema(db);
}
```

#### 8.5.2.18. truncatetext.js Filter

The truncatetext.js filter truncates a MySQL BLOB field.

| Pre-configured filter name | (not configured) |
| JavaScript Filter File | tungsten-replicator/samples/extensions/javascript/truncatetext.js |
| Property prefix | |
| Stage compatibility | binlog-to-q, q-to-dbms |
| tpm Option compatibility | --svc-extractor-filters [249], --svc-extractor-filters [249] |
| Data compatibility | Row events only |
Replication Filters

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>length</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:

replicator.filter.truncate.text=com.continuent.tungsten.replicator.filter.JavaScriptFilter
replicator.filter.truncate.text.script=${replicator.home.dir}/samples/extensions/javascript/truncate.text.js
replicator.filter.truncate.text.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` for a serialized BLOB value:

```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);
        }
    }
}
```

### 8.5.2.19. zerodate2null.js Filter

The `zerodate2null.js` 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/samples/extensions/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><code>tpm</code> Option compatibility</td>
<td>--svc-applier-filters [249]</td>
</tr>
<tr>
<td>Data compatibility</td>
<td>Row events only</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 type (`Types_TIMESTAMP`).

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.

```javascript
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 == Types_TIMESTAMP))
        {
            for (row = 0; row < columnValues.size(); row++)
            {
                values = columnValues.get(row);
                value = values.get(c);
                if (value.getValue() == 0)
                {
                    value.setValueNull();
                }
            }
        }
    }
}
```
Chapter 9. Performance and Tuning

To help improve the performance of Continuent Tungsten, a number of guides and tuning techniques are provided in this chapter. This may involve parameters entirely within Continuent Tungsten, or changes and modifications to the parameters within the OS.

Tuning related to the Tungsten Replicator functionality

• Section 9.1, "Block Commit" — Increasing performance of replication solutions making use of block commit.

Tuning related to the network performance

• Section 9.2, "Improving Network Performance" — Increasing performance of networking between components by tuning OS parameters.

9.1. Block Commit

Introduced in 2.0.1. 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 `blockCommitRowCount`)
• When the commit timer reaches the specified commit interval (set by `blockCommitInterval`)

The default operation is for block commitments 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.

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.

```
shell> /tools/tpm update alpha \
   --repl-svc-applier-block-commit-size=20 \
   --repl-svc-applier-block-commit-interval=10s
```

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=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 Vertica in [Tungsten Replicator 2.2 Manual], larger block commit sizes and intervals may improve performance during the batch loading process:

```
shell> /tools/tpm update alpha \
   --repl-svc-applier-block-commit-size=100000 \
   --repl-svc-applier-block-commit-interval=60s
```

This sets a large block commit size and interval enabling large batch loading.
9.1.1. 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
applyTime          : 4656.231
averageBlockSize   : 0.500
committed           : false
commits            : 10
currentBlockSize   : 0
currentLastEventId : mysql-bin.000015:0000000000001117;0
currentLastFragno  : 0
currentLastSeqno   : 5271
eventCount         : 5
extractTime        : 0.394
filterTime         : 0.017
lastCommittedBlockSize: 1
lastCommittedBlockTime: 0.033
otherTime          : 0.001
state              : extract
taskId             : 0
Finished status command (tasks)...
```

9.2. 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.

When using the Connector and client applications, improving the network performance will aid the overall performance of your application during both the client to connector, and connector to MySQL server connectivity.

The following network parameters should be configured within your `/etc/sysctl.conf` and can safely applied to all the hosts within your cluster deployments:

```
# Increase size of file handles and inode cache
fs.file-max = 2097152

# tell the kernel how many TCP sockets that are not attached to any user file handle to maintain. In case this number is exceeded, orphaned connections are immediately reset and a warning is printed.
net.ipv4.tcp_max_orphans = 60000

# Do not cache metrics on closing connections
net.ipv4.tcp_no_metrics_save = 1

# Turn on window scaling which can enlarge the transfer window:
net.ipv4.tcp_window_scaling = 1

# Enable timestamps as defined in RFC1323:
net.ipv4.tcp_timestamps = 1

# Enable select acknowledgments:
net.ipv4.tcp_sack = 1

# Maximum number of remembered connection requests, which did not yet receive an acknowledgment from connecting client.
net.ipv4.tcp_max_syn_backlog = 10240

# recommended default congestion control is htcp
net.ipv4.tcp_congestion_control = htcp

# recommended for hosts with jumbo frames enabled
net.ipv4.tcp_mtu_probing = 1

# Number of times SYNACKs for passive TCP connection.
net.ipv4.tcp_synack_retries = 2

# Allowed local port range
net.ipv4.ip_local_port_range = 1024 65535
```
Performance and Tuning

# Protect Against TCP Time-Wait
net.ipv4.tcp_rfc1337 = 1

# Decrease the time default value for tcp_fin_timeout connection
net.ipv4.tcp_fin_timeout = 15

# Increase number of incoming connections
# somaxconn defines the number of request_sock structures
# allocated per each listen call. The
# queue is persistent through the life of the listen socket.
net.core.somaxconn = 1024

# Increase number of incoming connections backlog queue
# Sets the maximum number of packets, queued on the INPUT
# side, when the interface receives packets faster than
# kernel can process them.
net.core.netdev_max_backlog = 65536

# Increase the maximum amount of option memory buffers
net.core.optmem_max = 25165824

# Increase the maximum total buffer-space allocatable
# This is measured in units of pages (4096 bytes)
net.ipv4.tcp_mem = 65536 131072 262144
net.ipv4.udp_mem = 65536 131072 262144

### Set the max OS send buffer size (wmem) and receive buffer
# size (rmem) to 12 MB for queues on all protocols. In other
# words set the amount of memory that is allocated for each
# TCP socket when it is opened or created while transferring files

# Default Socket Receive Buffer
net.core.rmem_default = 25165824

# Maximum Socket Receive Buffer
net.core.rmem_max = 25165824

# Increase the read-buffer space allocatable (minimum size,
# initial size, and maximum size in bytes)
net.ipv4.tcp_rmem = 20480 12582912 25165824
net.ipv4.udp_rmem_min = 16384

# Default Socket Send Buffer
net.core.wmem_default = 25165824

# Maximum Socket Send Buffer
net.core.wmem_max = 25165824

# Increase the write-buffer-space allocatable
net.ipv4.tcp_wmem = 20480 12582912 25165824
net.ipv4.udp_wmem_min = 16384

# Increase the tcp-time-wait buckets pool size to prevent simple DOS attacks
net.ipv4.tcp_max_tw_buckets = 1440000
net.ipv4.tcp_tw_recycle = 1
net.ipv4.tcp_tw_reuse = 1
Appendix A. 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 A.1, "Contacting Support".
- Error/Cause/Solution guidance on specific issues and error messages, and how the reason can be identified and resolved, see Section A.2, "Error/Cause/Solution".
- Additional troubleshooting for general systems and operational issues.

A.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.

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.

Generating Diagnostic Information

To aid in the diagnosis of issues, a copy of the logs and diagnostic information will help the support team to identify and trace the problem. There are two methods of providing this information:

- Using `tpm diag`

  The `tpm diag` command will collect the logs and configuration information from the active installation and generate a Zip file with the diagnostic information for all hosts within it. The command should be executed from the staging directory. Use `tpm query staging` to determine this directory:

  ```
  shell> tpm query staging
  tungsten@host1:/home/tungsten/continuent-tungsten-2.0.5-3
  shell> cd /home/tungsten/continuent-tungsten-2.0.5-3
  shell> ./tools/tpm diag
  ```

  The process will create a file called `tungsten-diag-2014-03-20-10-21-29.zip`, with the corresponding date and time information replaced. This file should be included in the reported support issue as an attachment.

  For a staging directory installation, `tpm diag` will collect together all of the information from each of the configured hosts in the cluster. For an INI file based installation, `tpm diag` will connect to all configured hosts if `ssh` is available. If a warning that `ssh` is not available is generated, `tpm diag` must be run individually on each host in the cluster.

- Manually Collecting Logs

  If `tpm diag` cannot be used, or fails to return all the information, the information can be collected manually:

  1. Run `tpm reverse` on all the hosts in the cluster:

     ```
     shell> tpm reverse
     ```

  2. Collect the logs from each host. Logs are available within the `service_logs` directory. This contains symbolic links to the actual log files. The original files can be included within a `tar` archive by using the `-h` option. For example:

     ```
     shell> cd /opt/continuent
     shell> tar zcfh host1-logs.tar.gz ./service_logs
     ```

     The `tpm reverse` and log archives can then be submitted as attachments with the support query.

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.
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.

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.

A.2. Error/Cause/Solution

A.2.1. Connector shows errors with "java.net.SocketException: Broken pipe"

Condition

When using DirectReads, the connector reports errors with a broken pipe.

Causes

• The most likely culprit for this error is that the wait_timeout and/or interactive_timeout is too low. This causes a problem because pooled connections get timeouts and are closed by the MySQL server.

Rectifications

• Change the configuration for your MySQL server (in my.cnf) to increase these timeouts.

A.2.2. MySQL is incorrectly configured

Condition

The configuration of MySQL was wrong; it included autocommit=0 and the wrong server-id

Causes

• Pre-requisites were not followed correctly.

Rectifications

• Edit my.cnf and clean up. Restart MySQL if possible. Alternatively, set manually:

```
mysql> set GLOBAL autocommit=1;
Query OK, 0 rows affected (0.00 sec)
mysql> set GLOBAL server_id=2;
Query OK, 0 rows affected (0.01 sec)
```

A.2.3. MySQLExtractException: unknown data type 0

Condition

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.

A.2.4. `cctrl` hangs

*Last Updated: 2013-11-01*

**Condition**

cctrl hangs

**Causes**

- Within Continuent Tungsten 1.5.3 there is a known issue related to how the managers handle failed network connections. It’s an indication that there was a network issue at some point.

**Rectifications**

- The temporary workaround is to put the cluster into maintenance mode, stop all managers, wait 10 seconds, then start them backup again.

**More Information**

Section 2.12.3, "Restarting the Manager Service"

A.2.5. ERROR 1580 (HY000) at line 5093787: You cannot 'DROP' a log table if logging is enabled

*Last Updated: 2013-11-01*

**Condition**

Loading a `mysqldump` into a MySQL server from a backup/restore fails.

**Causes**

- This appears to be a bug in MySQL that causes `mysqldump` loads to fail.

**Rectifications**

- You should be able to import the dump by switching off the slow query log globally before running the import:

  ```sql
  mysql> SET GLOBAL slow_query_log=0
  ```

A.2.6. Number of connections exceeded for MySQL

*Last Updated: 2013-07-17*

**Condition**

Connections to MySQL through the connector report that there are too many connections open.

**Causes**

- The maximum number of connections supported by MySQL is dependent on the available memory. If the available memory is exceeded, then the maximum number of connections may be reached, which in turn will lead to errors connecting to MySQL, either directly or through the connector.

**Rectifications**
Troubleshooting

- The maximum number of supported connections for MySQL can be determined using the following query:

```
mysql> SELECT ( @@key_buffer_size + @@query_cache_size + @@tmp_table_size + 
@innodb_buffer_pool_size + @@innodb_additional_mem_pool_size + 
@@innodb_log_buffer_size + @@max_connections * 
@@read_buffer_size + @@read_rnd_buffer_size + @@sort_buffer_size + 
@@join_buffer_size + @@binlog_cache_size + @@thread_stack ) / 1073741824 AS MAX_MEMORY_GB;
```

If the size of this value is greater than the available memory on the host running MySQL, the number of connections configured through the `max_connections` parameter should be modified.

More Information

Chapter 4, Operations Guide

A.2.7. Backup/Restore is not being my host back to normal

Last Updated: 2013-11-01

Condition

A backup/restore were performed as requested, but the host is still not coming up.

Causes

- When you backup a node, the backup is stored on that physical server. The correct backup file from an active server should be used on the host being restored.

Rectifications

- You can use that backup to restore another server in two ways:
  - If the backup directory is shared between servers using NFS or a clustered file system, the commands will work like you tried.
  - You must copy the backup files between nodes. See Section 4.10.3, "Restoring from Another Slave" for instructions on that.

A.2.8. Too many open processes or files

Last Updated: 2013-10-09

Condition

The operating system or environment reports that the tungsten or designated Continuent Tungsten 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) 79
virtual memory (kbytes, -v) unlimited
```

If the figures reported are less than the recommended settings, see Section C.2.1, "Creating the User Environment" for guidance on how these values should be changed.

More Information

Section C.2.1, “Creating the User Environment”
A.2.9. ERROR 2013 (HY000) at line 583: Lost connection to MySQL server during query

Last Updated: 2013-11-01

Condition
Client was disconnected during a query with the error number.

Causes
• Usually this means that the MySQL server has closed the connection or the server has restarted. The exact cause will be more difficult to determine.

Rectifications
• We need a bit more information to provide assistance.
  1. Were you connected through the Tungsten Connector?
  2. Did anything else happen on the servers?
  3. If you were connected through the Tungsten Connector, please upload the tungsten-connector/log/connector.log file from the server you were connected to.

A.2.10. Unable to update the configuration of an installed directory

Last Updated: 2013-08-07

Condition
Running an update or configuration with tpm returns the error ‘Unable to update the configuration of an installed directory’

Causes
• Updates to the configuration of a running cluster must be performed from the staging directory where Continuent Tungsten was originally installed.

Rectifications
• Change to the staging directory and perform the necessary commands with tpm. To determine the staging directory, use:
  
  ```shell
  shell> tpm query staging
  ```

  Then change to the staging directory and perform the updates:
  
  ```shell
  shell> ./tools/tpm configure ....
  ```

More Information
Chapter 2, Deployment

A.2.11. cctrl reports MANAGER(state=STOPPED)

Last Updated: 2013-11-01

Condition
cctrl reports the status for the manager as MANAGER(state=STOPPED)

Causes
• The manager has stopped running, possibly due to a fault or error state.

Rectifications
• Restart the manager process on this server is not running. You can start it by running:
  
  ```shell
  shell> manager start
  ```

Or:
A.2.12. Replicator fails to connect after updating password

Condition
Tungsten Replicator fails to connect after changing the `tungsten` user password

Causes
• The most likely cause is that the configuration within `~/.my.cnf` was forcing a connection to the cluster as `tungsten` user, and user change may have only been made on one host and not replicated to the other MySQL servers.

Rectifications
• First, update the credentials in `~/.my.cnf` and ensure you can connect to all the slaves with the updated credentials.

Also check that `tpm` has been configured with the right password and that all servers have the right information. Errors such as:

```
ERROR >> host1 >> Unable to connect to the MySQL server using »
Tungsten@host1:3306 (WITH PASSWORD) (MySQLLoginCheck)
```

Indicate that the password may not have been replicated properly. Check the following:

1. Check the user configuration information within each MySQL server and compare the values:
   ```
   mysql> select * from mysql.user where user='tungsten';
   ```

2. For any node that is not up to date, update the password manually:
   ```
   shell> mysql -u root -ppassword -P 3306 -h host1
   mysql> UPDATE `mysql`.`user` SET Password=PASSWORD('secret') WHERE User='tungsten';
   mysql> flush privileges;
   ```

3. Update the `tpm` and Continuent Tungsten configuration:
   ```
   shell> ./tools/tpm update alpha --datasource-password=secret
   ```

4. Restart the replicators:
   ```
   shell> replicator restart
   ```

Then put the replicators offline/online to refresh the configuration:
```
[LOGICAL] /alpha > datasource host1 offlineDataSource 'host1@alpha' is now OFFLINE[LOGICAL] /alpha > datasource host1 onlineSetting server for data source 'host1' to READ-ONLY```

A.2.13. ERROR 1010 (HY000) at line 5094506: Error dropping database (can't rmdir './mysql-bin/', errno: 17)

Condition
Loading a `mysqldump` into a MySQL server from a backup/restore fails.

Causes
• The problem may be that your MySQL binary logs are in a subdirectory of your MySQL data directory, causing MySQL to view them as a schema.

Rectifications
• Possible steps to resolution:
1. Modify the dump file so it isn’t trying to drop a schema named after the bin log directory
2. Update the mysql configuration so the bin logs aren’t in a directory in the data dir. mysql sees all directories in the data dir as a schema

A.2.14. Triggers not firing correctly on slave

Last Updated: 2013-11-01

Condition

Newly created triggers are not firing when executed

Causes

• If a new user (definer) was used to create the triggers, they may fail to be executed, raising the following warning in the logs:

```
INFO | jvm 1 | 2013/10/16 04:21:33 | WARNING: Could not execute query »
org.drizzle.jdbc.internal.common.query.DrizzleQuery@60dc4c81: The »
MySQL server is running with the --read-only option so it cannot »
execute this statement
INFO | jvm 1 | 2013/10/16 04:21:33 | 2013-10-16 04:21:33,208 ERROR »
replicator.pipelineSingleNodeStatementTask [q-to-dm] Event »
application failed: seqno=524545571 fragno=0 message=java.sql.SQLException: »
Statement failed on slave but succeeded on master
INFO | jvm 1 | 2013/10/16 04:21:33 | com.continuent.tungsten.replicator.applier.ApplierException: »
java.sql.SQLException: Statement failed on slave but succeeded on master
```

This is an indication that the new definer does not have the required SUPER privilege and that a trigger is failing to run.

Rectifications

• In order to fix this issue, the new definer should be given the SUPER privilege on each server and then replication should be restarted. The SUPER privilege allows the user to run a statement on a slave server where the read_only flag has been turned on. If necessary, the scope of the privilege can be restricted to an individual schema. The GRANT statement should be done on every database server, while the shun and recover should only be done on the slaves.

```
mysql> grant SUPER on *.* to user;
mysql> flush privileges;
```

Within cctrl:

```
cctrl> datasource hostname shun;
cctrl> datasource hostname recover;
```

You should continue to review the tungsten-replicator/log/trepsvc.log file to see what log messages are being written there. It appears that replication is still failing and it is probably related to the same issue. If you want us to review logs to interpret the results for you, you can upload the log file here and someone will look at it.

More Information

Section A.3.1, “Triggers”

A.2.15. Event application failed: seqno=20725782 fragno=0 message=java.sql.SQLDataException: Data too long for column 'eventid' at row 1

Last Updated: 2013-11-01

Condition

Event application failed: seqno=20725782 fragno=0 message=java.sql.SQLDataException: Data too long for column 'eventid' at row 1

Causes

• The issue is that the eventid column in tungsten.heartbeat is shorter than tungsten.eventid. You could do an alter on the master to extend that column and let that replicate out. The column sizes match in the next version.

Rectifications

• The tables must be updated:

```
mysql> ALTER TABLE `heartbeat` CHANGE `eventid` `eventid` VARCHAR(128) 
```
This will update the tables from the following structure:

```
mysql> SHOW CREATE TABLE tungsten.heartbeat;
heartbeats | CREATE TABLE `heartbeat` (
    `id` bigint(20) NOT NULL DEFAULT '0',
    `seqno` bigint(20) DEFAULT NULL,
    `eventid` varchar(32) DEFAULT NULL,
    `source_tstamp` timestamp NULL DEFAULT NULL,
    `target_tstamp` timestamp NULL DEFAULT NULL,
    `lag_millis` bigint(20) DEFAULT NULL,
    `salt` bigint(20) DEFAULT NULL,
    `name` varchar(128) DEFAULT NULL,
    PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

```
mysql> SHOW CREATE TABLE tungsten.trep_commit_seqno;
trep_commit_seqno | CREATE TABLE `trep_commit_seqno` (
    `seqno` bigint(20) DEFAULT NULL,
    `fragno` smallint(6) DEFAULT NULL,
    `last_frag` char(1) DEFAULT NULL,
    `source_id` varchar(128) DEFAULT NULL,
    `epoch_number` bigint(20) DEFAULT NULL,
    `eventid` varchar(128) DEFAULT NULL,
    `applied_latency` int(11) DEFAULT NULL,
    `update_timestamp` timestamp NULL DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

Afterwards the table will be formatted:

```
heartbeats | CREATE TABLE `heartbeat` (
    `id` bigint(20) NOT NULL DEFAULT '0',
    `seqno` bigint(20) DEFAULT NULL,
    `eventid` varchar(128) DEFAULT NULL,
    `source_tstamp` timestamp NULL DEFAULT NULL,
    `target_tstamp` timestamp NULL DEFAULT NULL,
    `lag_millis` bigint(20) DEFAULT NULL,
    `salt` bigint(20) DEFAULT NULL,
    `name` varchar(128) DEFAULT NULL,
    PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

A.2.16. 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

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.

A.2.17. 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

Replication fails due to an incorrect SQL mode, INVALID_DATES being applied for a specific transaction.
A.2.18. Starting replication after performing a restore because of an invalid restart sequence number

Last Updated: 2013-11-01

Condition
Starting replication fails because of an invalid restart sequence number. Checking the sequence number, `trep_commit_seqno` shows an empty or invalid table contents:

```sql
mysql> select * from tungsten.trep_commit_seqno;
+-------+--------+-----------+-----------+--------------+---------+-----------------+---------------------+
<table>
<thead>
<tr>
<th>seqno</th>
<th>fragno</th>
<th>last_frag</th>
<th>source_id</th>
<th>epoch_number</th>
<th>eventid</th>
<th>applied_latency</th>
<th>update_timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
<td>-1</td>
<td>2013-10-27 23:44:05</td>
</tr>
</tbody>
</table>
+-------+--------+-----------+-----------+--------------+---------+-----------------+---------------------+
1 row in set (0.00 sec)
```

Causes
- The restore may have failed to correctly restore the `tungsten` tables.

Rectifications
- Retry the restore process, making sure the replicator is stopped and there are no updates to the table taking place:

  1. `shell> replicator stop` (ensure no replicator processes are running)
  2. `shell> tpm reset-thl` (ensure the `/opt/continuent/thl` directory is empty)
  3. Restore the backup using whatever backup/restore tool you used.
  4. Check that `trep_commit_seqno` has a valid restart position in it
  5. `shell> replicator start`

A.2.19. Replicator reports an Out of Memory error

Last Updated: 2013-11-01

Condition
Replicator reports an Out of Memory error

Causes
- The configured memory sizes within the replicator are too small for the data being replicated and applied.

Rectifications
- Raise the `java.maxmemory` parameter, for example to 3072 (specified in megabytes) within `wrapper.conf` and restart the replicator.

More Information
Section F.4, "Memory Tuning and Performance"

A.2.20. 'subscription exists' when setting up CDC on Oracle

Last Updated: 2014-04-24

Condition
When running `setupCDC.sh` an error regarding an existing subscription is received and `setupCDC.sh` fails to complete.

**Causes**

- This error indicates that `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 `cleanup_cdc_tables.sql` file can be used to cleanup an existing CDC installation. See CDC Cleanup and Correction in [Tungsten Replicator 2.2 Manual].

### A.2.21. Replication latency very high

*Last Updated: 2013-11-01*

**Condition**

The latency of updates on the slaves is very high.

**Causes**

- First the reason and location of the delay should be identified. It is possible for replication data to have been replicated quickly, but applying the data changes is taking a long time. Using row-based replication may increase the latency due to the increased quantity of data that must be transferred.

**Rectifications**

- Check the replication format:

  ```
  shell> grep binlog_format /etc/my.cnf
  binlog_format=ROW
  ```

  Slow slaves can be the cause, but it may require some configuration changes.

### A.2.22. trepctl status hangs

*Last Updated: 2013-11-01*

**Condition**

`trepctl status` hangs at the end of the output after a "cannot fork" error due.

**Causes**

- This can be caused by THL corruption

**Rectifications**

- You should recreate the THL files on the slave. This can be achieved by deleting the existing THL files, which will cause the slave replicator to download all of the THL data from the master again:

  ```
  shell> replicator stop
  shell> cd /opt/continuent
  shell> mv thl thl.old;
  shell> mkdir thl
  shell> replicator start
  shell> trepctl status
  ```

### A.3. Known Issues

#### A.3.1. Triggers

Tungsten Replicator does not automatically shut off triggers on slaves. This creates problems on slaves when using row-based replication (RBR) as the trigger will run twice. Tungsten cannot do this because the setting required to do so is not available to MySQL client applications. Typical symptoms are duplicate key errors, though other problems may appear. Consider the following fixes:

- Drop triggers on slaves. This is practical in fan-in for reporting or other cases where you do not need to failover to the slave at a later time.
• Create an `is_master()` function that triggers can use to decide whether they are on the master or slave.

• Use statement replication. Beware, however, that even in this case you may find problems with triggers and auto-increment keys.

   The `is_master()` approach is simple to implement. First, create a function like the following that returns 1 if we are using the Tungsten user, as would be the case on a slave.

```plaintext
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.

```plaintext
delimiter //
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.

A.4. Troubleshooting Timeouts

A.5. Troubleshooting Backups

• Operating system command failed

   Backup directory does not exist.

```plaintext
INFO | jvm 1 | 2013/05/21 09:36:47 | Process timed out: false
INFO | jvm 1 | 2013/05/21 09:36:47 | Process exception null
INFO | jvm 1 | 2013/05/21 09:36:47 | Process stderr: Error: »
   The directory '/opt/continuent/backups/xtrabackup' is not writeable
```

• Backup Retention

A.6. Running Out of Diskspace

```plaintext
... pendingError : Event application failed: seqno=156847 »
   frame=0 message=Unable to store event: seqno=156847
pendingErrorCode : NONE
pendingErrorEventId : mysql-bin.000025:0000000024735754:0
pendingErrorSeqno : 156847
pendingExceptionMessage: Unable to store event: seqno=156847
...
```

The above indicates that the THL information could not be stored on disk. To recover from this error, make space available on the disk, or move the THL files to a different device with more space, then set the replicator service online again.

For more information on moving THL files to a different disk, see Section E.1.4.2, “Moving the THL File Location”; for information on moving the backup file location, see Section E.1.1.4, “Relocating Backup Storage”.

A.7. Troubleshooting SSH and tpm

When executing `tpm`, `ssh` is used to connect and install the software on other hosts in the cluster. If this fails, and the public key information is correct, there are a number of operations and settings that can be checked. Ensure that you have followed the Section C.2.2.2, “SSH Configuration” instructions.

• The most likely representation of this error will be when executing `tpm` during a deployment:

```
Error:   >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```
Troubleshooting

Validation failed
#####################################################################
Errors for host1
ERROR >> host1 >> Unable to SSH to host1 as root. (SSHLoginCheck)
Ensure that the host is running and that you can login as root via SSH using key authentication
tungsten-configure.log shows:
2012-05-23T11:10:37+02:00 DEBUG >> Execute `whoami` on host1 as root
2012-05-23T11:10:38+02:00 DEBUG >> RC: 0, Result: stdin: is not a tty

Try running the following command:

shell> ssh tungsten@host1 sudo whoami

If the SSH and sudo configurations have been configured correctly, it should return root. Any other value indicates a failure to configure the prerequisites properly.

• Check that none of the profile scripts (.profile, .bash_profile, .bashrc, etc.) do not contain a call to mesg n. This may fool the non-interactive ssh call; the call to this command should be changed to only be executed on interactive shells:

if "tty" -s; then
  mesg n
fi

• Check that firewalls and/or antivirus software are not blocking or preventing connectivity on port 22.

if ssh has been enabled on a non-standard port, use the --net-ssh-option=port 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.

A.8. Troubleshooting Data Differences

A.9. Comparing Table Data

The Percona Toolkit includes a tool called pt-table-checksum that enables you to compare databases on different databases using a checksum comparison. This can be executed by running the checksum generation process on the master:

shell> pt-table-checksum --set-vars innodb_lock_wait_timeout=500 \
  --recursion-method=none \n  --ignore-databases=mysql \n  --ignore-databases-regex=tungsten* \n  h=localhost,u=tungsten,p=secret

Using MySQL, the following statement must then be executed to check the checksums generated on the master:

mysql> <userinput>SELECT db, tbl, SUM(this_cnt) AS total_rows, COUNT(*) AS chunks \n  FROM percona.checksums WHERE ( master_cnt <> this_cnt OR master_crc <> this_crc OR ISNULL(master_crc) <> ISNULL(this_crc)) GROUP BY db, tbl;</userinput>

Any differences will be reported and will need to manually corrected.

A.10. Troubleshooting Memory Usage
Appendix B. Release Notes

B.1. Continuent Tungsten 2.0.5 GA (24 Dec 2014)

Continuent Tungsten 2.0.5 is a bugfix release that contains critical improvements to the handling of times, dates, and timestamp values between servers, including during daylight savings time switches.

Improvements, new features and functionality

• Installation and Deployment

• An issue was discovered that altered the way different date and time values were extracted, stored in THL, and applied into target databases. The issue was related to the way the value was stored; the data was not normalized within Continuent Tungsten during replication, particularly if different timezones were used and applied across the replication deployment.

Examples of the behaviour include:

• MySQL converts `TIMESTAMP` values in statements to UTC. Tungsten did not replicate the master time zone, which meant that replicated statements would generate different `TIMESTAMP` values when replicated to a server with a different time zone from the master.

• MySQL `TIMESTAMP` values are stored as UTC, which means that row changes are extracted in UTC. Tungsten did not set the Java VM or MySQL session time zone to UTC when applying such changes, which could result in inconsistent values being applied to replicas.

• Changes between standard and daylight savings time (DST) result in a short period in which master DBMS servers have a different time zone from replicas. This resulted in errors in applying time-related data generated at the time of the switch.

• Heterogeneous replication, for example from relational DBMS like MySQL to data warehouses, would result in unexpected conversions to time-related data, again due to inconsistencies in time zones.

The replication has now been updated to normalize date and time values into UTC throughout the replication topology, including within the wrapper Java processes, databases and when storing the information in THL.

• Replicator processes now default to UTC internally by setting the Java VM default time zone to UTC. This default can be changed by setting the replicator.time_zone property in the replicator services.properties file but is not recommended other than for problem diagnosis or specialized testing.

• Replicas store a time zone on statements and row changes extracted from MySQL.

• Replicators use UTC as the session time zone when applying to MySQL replicas.

• Replicators similarly default to UTC when applying transactions to data warehouses like Hadoop, Vertica, or Amazon Redshift.

• The `thl` utility prints time-related data using the default GMT time zone. This can be altered using the `-timezone` option.

Best Practices

We recommend the following steps to ensure successful replication of time-related data.

• Standardize all DBMS server and host time zones to UTC. This minimizes time zone inconsistencies between applications and data stores. The recommendation is particularly important when replicating between different DBMS types, such as MySQL to Hadoop.

• Use the default time zone settings for Tungsten replicator. Do not change the time zones unless specifically recommended by VMware support.

• If you cannot standardize on UTC at least ensure that time zones are set consistently on all hosts and applications.

Arbitrary time zone settings create a number of corner cases for database management beyond replication. Standardizing on UTC helps minimize them, hence is strongly recommended.

Upgrade from Older Replicator Versions

New Tungsten replicators tag THL records with an option to show that the transaction was extracted from a time zone-aware replicator. If a replicator sees that this property is not available, it will automatically switch to the older behavior when applying such transactions to MySQL replicas. This ensures that there is as simple process to upgrade from older replicator versions, which is especially important for Continuent Tungsten clusters.

There are two ways to upgrade a replication topology that extracts from MySQL to the new, time zone-aware behavior.
• Put the master replicator offline, wait for slaves to catch up fully, then upgrade all replicators at once.

• Upgrade slave replicators first, then upgrade the master. If the replicators are running in a Continuent Tungsten cluster, you must put the cluster in maintenance mode during the upgrade to prevent master failover.

**Important**

You should not upgrade a master Tungsten Replicator before the slave replicas. This can generate transactions that may not be correctly applied by the slaves, since they are not time zone-aware.

For more information, see Section F.3, "Replication of Date/Time Values".

### B.2. Continuent Tungsten 2.0.4 GA (9 Sep 2014)

This is a recommended release for all customers as it contains important updates and improvements to the stability of the manager component, specifically with respect to stalls and memory usage that would cause manager failures.

We recommend Java 7 for all Continuent Tungsten 2.0 installations. Continuent are aware of issues within Java 6 that cause memory leaks which may lead to excessive memory usage within the manager. This can cause the manager to run out of memory and restart, without affecting the operation of the dataservice. These problems do not exist within Java 7.

**Improvements, new features and functionality**

• **Tungsten Manager**

  • **Tungsten Manager: Improved monitoring fault-tolerance**

    Under normal operating conditions, the Tungsten Manager on each DB server host will monitor the local Tungsten Replicator and the database server running on that host and relay the monitoring information thus collected to the other Tungsten Managers in the cluster. In previous releases, Continuent Tungsten was even able to continue to monitor database servers even if a manager on a given DB server node was not running.

    With this release, this functionality has been generalized to handle the monitoring of both database servers and Tungsten replication such that any time a Tungsten Manager is not running on a given DB server host, the remaining Tungsten Managers in the cluster will take over the monitoring activities for both database servers and Tungsten Replicators until the manager on that host resumes operations. This activity takes place automatically and does not require any special configuration or intervention from an administrator.

    The new functionality means that if you have configured Tungsten to fence replication failures and stoppages, and you stop all Tungsten services on a given node, the rest of the cluster will respond by fencing the associated data source to an OFFLINE or FAILED state.

    Full recovery of a failed node requires that a Tungsten Manager be running on the node.

• **Tungsten Connector/Tungsten Manager: Full support for 'relative latency'**

  Support for the use and display of the `relativeLatency` has been expanded and improved. By default, absolute latency is used by the cluster to determine the configuration.

  When relative latency is used, the difference between the last commit time and the current time is displayed. This will show an increasing latency even on long running transactions, or in the event of a stalled replicator. To enable relative latency, use the `--use-relative-latency=true` option to `tpm` during configuration.

  The following changes to the operation of Continuent Tungsten have been added to this release when the use of relative latency is enabled:

  • The output of `SHOW SLAVE STATUS` has been updated to show the `Seconds_Behind_Master` value.

  • `cctrl` will output a new field, `relative`, showing the relative latency value.

  • The Tungsten Connector will use the value when the `maxAppliedLatency` option is used in the connection string to determine whether to route a connection to a master or a slave.

    For more information, see Section 4.1.1, "Latency or Relative Latency Display".

• **Tungsten Manager: Automated Data Source Fencing Due to Replication Faults**

  Continuent Tungsten can now be configured to effectively isolate data sources for which replication has stopped or exhibits an error condition. See the updated documentation on Section 4.4, "Replicator Fencing" for further information.
Release Notes

Issues: TUC-2240

For more information, see Section 4.4, “Replicator Fencing”.

Bug Fixes

• Installation and Deployment

• The tpm command has been updated to support updated fencing mechanisms.

  Issues: TUC-2245

• During an upgrade procedure, the process would mistake active witnesses for passive ones.

  Issues: TUC-2280

• During an update using tpm, the replicator could end up in the offline state.

  Issues: TUC-2282

• When performing an update, particularly in environments such as Multi-Site, Multi-Master, the tpm command could fail to update the cluster correctly. This could leave the cluster in a diminished state, or fail to upgrade all the components. The tpm command has been updated as follows:

  • tpm will no longer attempt to upgrade a Tungsten Replicator™ with a Continuent Tungsten™ distribution, and vice versa.

  • When installing Tungsten Replicator™, and the $CONTINUENT_PROFILES variable has been set, tpm will fail, warning that the $REPLICATOR_PROFILES variable should be set instead.

  Issues: TUC-2288, TUC-2292

• Tungsten Connector

• When changing connector properties, and reloading the configuration, the updated values would not be updated.

• When using mysqldump with option --flush-logs, the connector would fail with an Unsupported command error.

  Issues: TUC-2209

• When the option showRelativeSlaveStatus=true has been specified, the behavior of the connector for checking of latency with read/write splitting would not be used, instead the appliedLatency figure would be used instead.

  Issues: TUC-2243

• The connection.close.idle.timeout would fail to be taken into account when the connector was running in bridge mode.

  Issues: TUC-2255

• When the connector was running in bridge mode, and the connection was killed, the connections would not be correctly closed.

  Issues: TUC-2261

• The Connector SmartScale would fail to round-robin through slaves when there was no discernable load on the cluster to provide load performance metrics.

  Issues: TUC-2272

• SmartScale would wrongly load balance connections to a slave even during a switch operation.

  Issues: TUC-2273

• The connector would update the high water setting before and after a write connection was used, creating additional overhead for connections, generating additional query overhead.

  Issues: TUC-2277

• When using SmartScale, automatic sessions could be unnecessarily closed upon disconnection, causing slaves to miss valid queries.

  Issues: TUC-2286

• Tungsten Manager
• The `checker.tungstenreplicator.properties` and `checker.mysqlserver.properties` files would fail to be created correctly on active witnesses.

  **Issues:** TUC-2250, TUC-2251

• The manager would fail to show the correct status for the replicator when getting status information by proxy.

  **Issues:** TUC-2254

• Under some conditions, the manager would shut down the router gateway due to an invalid membership alarm but would not restart the connector. This would cause all new connections to hang indefinitely.

  **Issues:** TUC-2278

• When performing a reset of the replicator service, recovery of the failed service would fail.

  **Issues:** TUC-2290

• Other Issues

  • The `check_tungsten.sh` script could fail to locate the `tungsten.cfg` or read the correct values from the file.

  **Issues:** TUC-2263

**B.3. Continuent Tungsten 2.0.3 GA (1 Aug 2014)**

This is a recommended release for all customers as it contains important updates and improvements to the stability of the manager component, specifically with respect to stalls and memory usage that would cause manager failures.

We recommend Java 7 for all Continuent Tungsten 2.0 installations. Continuent are aware of issues within Java 6 that cause memory leaks which may lead to excessive memory usage within the manager. This can cause the manager to run out of memory and restart, without affecting the operation of the dataservice. These problems do not exist within Java 7.

**Behavior Changes**

The following changes have been made to Continuent Tungsten and may affect existing scripts and integration tools. Any scripts or environment which make use of these tools should check and update for the new configuration:

• Within composite clusters, TCP/IP port 7 connectivity is now required between managers on each site to confirm availability.

**Known Issue**

The following issues may affect the operation of Continuent Tungsten and should be taken into account when deploying or updating to this release.

• The default behavior of the manager is to not fence a datasource for which a replicator has stopped or gone into an error state. This was implemented to prevent reducing the overall availability of the deployed service. There are cases and deployments where clusters should not operate with replicators in stopped or error states. This could be configurable by changing the following properties to `true` according to the master or slave role requirements:

  ```
  policy.fence.slaveReplicator=false
  policy.fence.masterReplicator=false
  ```

  If they are set to true, the manager should fence the datasource by setting it to a 'failed' state. When this happens, and the datasource is a master, failover will occur. If the datasource is a slave, the datasource will just stay in the failed state indefinitely or until the replicator is back in the online state, in which case the datasource will be recovered to online.

  At present the setting of these properties are not honored.

  **Issues:** TUC-2241

**Improvements, new features and functionality**

• Tungsten Connector

  • The default buffer sizes for the Section 5.3, "Using Bridge Mode" have been updated to 262144 (256KB).

**Bug Fixes**

• Installation and Deployment
• To ensure that the correct number of the managers and witnesses are configured within the system, tpm has been updated to check and identify potential issues with the configuration. The installation and checks operate as follows:
  • If there are an even number of members in the cluster (i.e. provided to --members [237] option):
    • If witnesses are provided through --witnesses [252], continue normally.
    • If witnesses are not provided through --witnesses [252], an error is thrown and installation stops.
  • If there are an odd number of members in the cluster (i.e. provided to --members [237] option):
    • If witnesses are provided through --witnesses [252], a warning is raised and the witness declaration is ignored.
    • If witnesses are not provided through --witnesses [252], installation continues as normal.

The number of members is calculated as follows:
  • Explicitly through the --members [237] option.
  • Implied, when --active-witnesses=false [231], then the list of hosts declared in --master [237] and --slaves [248].
  • Implied, when --active-witnesses=true [231], then the list of hosts declared in --master [237] and --slaves [248] and --witnesses [252].

Issues: TUC-2105
• If ping traffic was denied during installation, then installation could hang while the ping check was performed. A timeout has now been added to ensure that the operation completes successfully.
  Issues: TUC-2107

• Backup and Restore
  • When using xtrabackup 2.2.x, backups would fail if the innodb_log_file_size option within my.cnf was not specified. tpm has been updated to check the value and existence of this option during installation and to provide a warning if it is not set, or set to the default.
    Issues: TUC-2224

• Tungsten Connector
  • The connector will now re-connect to a MySQL server in the event that an opened connection is found closed between two requests (generally following a wait_timeout expiration).
    Issues: TUC-2163
  • When initially starting up, the connector would open a connection to the configured master to retrieve configuration information, but the connection would never be closed, leading to open unused connections.
    Issues: TUC-2166
  • The cluster status output by the tungsten cluster status command within a multi-site cluster would fail to display the correct states of different data sources when an entire data service was offline.
    Issues: TUC-2185
  • When the connector has been configured into read-only mode, for example using --application-readonly-port=9999 [220], the connector would mistakenly route statements starting set autocommit=0 to the master, instead of being routed to a slave.
    Issues: TUC-2198
  • When operating in bridge mode, the connector would retain the client connection when the server had closed the connection. The connector has been updated to close all client connections when the corresponding server connection is closed.
    Issues: TUC-2231

• Tungsten Manager
  • The manager could enter a situation where after switching a relay on one physical service, remote site relay is incorrectly reconfigured to point at the new relay. This has been corrected so that reconfiguration no longer occurs in this situation.
  Issues: TUC-2164
• Recovery from a composite cluster failover could create a composite split-brain situation.
  
  *Issues*: TUC-2178

• A statement of record (SOR) cluster would be unable to recover a failed dataservice.
  
  *Issues*: TUC-2194

• A composite datasource would not go into failsafe mode if all the managers within the cluster were stopped.
  
  *Issues*: TUC-2206

• If a composite datasource becomes isolated due to a network partition, the failed datasource would not go into failsafe mode correctly.
  
  *Issues*: TUC-2207

• If a witness became isolated from the rest of the cluster, the rules would not exclude the failed witness and this could lead to memory exhaustion.
  
  *Issues*: TUC-2214

• **Documentation**
  
  • The descriptions and definitions of the *archive* [89] and *standby* [89] roles has been clarified in the documentation.

  For more information, see Section 4.1.3, “Understanding Datasource Roles”.

• The documentation for the recovery of a multi-site multi-master installation has been updated to provide more information when covering.
  
  *Issues*: TUC-2175

  For more information, see Section 2.5.3.2, “Resetting a single dataservice”.

### B.4. Continuent Tungsten 2.0.2 GA (19 May 2014)

This is a recommended release for all customers as it contains important updates and improvements to the stability of the manager component, specifically with respect to stalls and memory usage that would cause manager failures.

In addition, we recommend Java 7 for all Continuent Tungsten 2.0 installations. Continuent are aware of issues within Java 6 that cause memory leaks which may lead to excessive memory usage within the manager. This can cause the manager to run out of memory and restart, without affecting the operation of the dataservice. These problems do not exist within Java 7.

#### Improvements, new features and functionality

• **Installation and Deployment**
  
  • The default Java garbage collection (GC) used within the Connector, Replicator and Manager has been reconfigured to use parallel garbage collection. The default GC could produce CPU starvation issues during execution.

  *Issues*: TUC-2101

• **Tungsten Connector**
  
  • Keep-alive functionality has been added to the Connector. When enabled, connections to the database server are kept alive, even when there is no client activity.

  *Issues*: TUC-2103

  For more information, see Section 5.5.5, “Connector Keepalive”.

#### Bug Fixes

• **Tungsten Manager**
  
  • A number of issues the memory management on the Manager service, particularly with respect to the included JGroups support have been rectified. These issues caused the manager to use increased amounts of memory that could lead to the manager to stall.

  • The embedded JGroups service, which manages the communication and management of the manager service has been updated to the latest version. This improves the stability of the service, and removes some of the memory leaks causing manager stalls.
Continuent Tungsten 2.0.2 includes the following changes from Tungsten Replicator 2.2.1

Behavior Changes

The following changes have been made to Tungsten Replicator and may affect existing scripts and integration tools. Any scripts or environment which make use of these tools should check and update for the new configuration:

• The `tpm` tool and configuration have been updated to support both older Oracle SIDs and the new JDBC URL format for Oracle service IDs. When configuring an Oracle service, use `--datasource-oracle-sid` for older service specifications, and `--datasource-oracle-service` [228] for newer JDBC URL installations.

Issues: 817

Improvements, new features and functionality

• Installation and Deployment
  • When using the `--enable-heterogeneous-master` option to `tpm`, the MySQL service is now checked to ensure that ROW-based replication has been enabled.

Issues: 834

• Command-line Tools
  • The `thl` command has been expanded to support an additional output format, `-specs`, which adds the field specifications for row-based THL output.

Issues: 801

For more information, see `thl list -specs Command`

• Oracle Replication
  • Templates have been added to the suite of DDL translation templates supported by `ddlscan` to support Oracle to MySQL replication. Two templates are included:
    • `ddl-oracle-mysql` provides standard translation of DDL when replicating from Oracle to MySQL
    • `ddl-oracle-mysql-pk-only` provides standard translation of DDL including automatic selection of a primary key from the available unique indexes if no explicit primary key is defined within Oracle DDL when replicating to MySQL

Issues: 787

• `ddlscan` has been updated to support parsing of a file containing a list of tables to be parsed for DDL information. The file should be formatted as a CSV file, but only the first argument, table name, will be extracted. Lines starting with a # (hash) character are ignored.

The file is in the same format as used by `setupCDC.sh`.

To use the file, supply the `-tableFile` parameter to the command.

Issues: 832

• Core Replicator
  • The replicator has been updated to support autorecovery from transient failures that would normally cause the replicator to go `OFFLINE` while in either the `ONLINE` or `GOING-ONLINE;SYNCHRONIZING` state. This enables the replicator to recover from errors such as MySQL restarts, or transient connection errors.

The period, number of attempted recovery operations, and the delay before a recovery is considered successful are configurable through individual properties.

Issues: 784

For more information, see Section 4.8, “Deploying Automatic Replicator Recovery”.

• The way `VARCHAR` values were stored and represented within the replicator has been updated which improves performance significantly.
Release Notes

Issues: 804
• If the binary logs for MySQL were flushed and purged (using `FLUSH LOGS` and `PURGE BINARY LOGS`), and then the replicator is restarted, the replicator would fail to identify and locate the newly created logs with an MySQLExtractException.

Issues: 851
• Documentation
  • The deployment and recovery procedures for Multi-site/Multi-master deployments have been documented.

For more information, see Section 2.5, “Deploying a MultiSite/MultiMaster Topology”.

Bug Fixes

Installation and Deployment
• `tpm` would mis-identify options that accepted true/false values, which could cause incorrect interpretations, or subsequent options on the command-line to be used as true/false indications.

Issues: 310
• Removing an existing parallel replication configuration using `tpm` would cause the replicator to fail due to a mismatch in the status table and current configuration.

Issues: 867

Command-line Tools
• The `tungsten_provision_slave` tool would fail to correctly re-provision a master within a fan-in or multi-master configuration. When re-provisioning, the service should be reset with `trepctl reset`.

Issues: 709
• Errors when executing `tungsten_provision_slave` that have been generated by the underlying `mysqldump` or `xtrabackup` are now redirected to STDOUT.

Issues: 802
• The `tungsten_provision_slave` tool would re-provision using a slave in a OFFLINE:ERROR state, even though this could create a second, invalid, slave deployment. Reprovisioning from a slave in the ERROR state is now blocked, unless the `-f` or `--force` option is used.

Issues: 860
For more information, see Section 7.10, “The `tungsten_provision_slave` Script”.

Oracle Replication
• Tuning for the CDC extraction from Oracle has been updated to support both a minimum sleep time parameter, `minSleepTime`, and the increment value used when increasing the sleep time between updates, `sleepAddition`.

Issues: 239
For more information, see Tuning CDC Extraction in [Tungsten Replicator 2.2 Manual].

• The URLs used for connecting to Oracle RAC SCAN addresses were not correct and were incompatible with non-RAC installations. The URL format has been updated to use a URL format that is compatible with both Oracle RAC and non-RAC installations.

Issues: 479

Core Replicator
• When a timeout occurred on the connection to MySQL for the channel assignment service (part of parallel applier), the replicator would go offline, rather than retrying the connection. The service has now been updated to retry the connection if a timeout occurs. The default reconnect timeout is 120 seconds.

Issues: 783
• A slave replicator would incorrectly set the restart sequence number when reading from a master if the slave THL directory was cleared. This would cause slave replicators to fail to restart correctly.

Issues: 794

• Unsigned integers are extracted from the source database in a non-platform independent method. This would cause the Oracle applier to incorrectly attempt to apply negative values in place of their unsigned equivalents. The Oracle applier has been updated to correctly translate these values for types identified as unsigned to the correct value. When viewing these values are viewed within the THL, they will still be identified as a negative value.

Issues: 798

For more information, see Section 7.7.1, "thl list Command".

• Replication would fail when processing binlog entries containing the statement \texttt{INSERT INTO ... WHERE...} when operating in mixed mode.

Issues: 807

• Filters

  • The \texttt{mysqlsessionsupport} filter would cause replication to fail when the default \texttt{thread_id} was set to -1, for example when \texttt{STRICT_ALL_TABLES} SQL mode had been enabled. The replicator has been updated to interpret -1 as 0 to prevent this error.

Issues: 821

  • The \texttt{rename} filter has been updated so that renaming of only the schema name for STATEMENT events. Previously, only ROW events would be renamed by the filter.

Issues: 842

B.5. Continuent Tungsten 2.0.1 GA (3 January 2014)

Important

The final approved build for Continuent Tungsten 2.0.1 is build 1003. Earlier builds do not have the full set of features and functionality, and includes a number of key fixes not in earlier builds of the same release. In particular, updated support for passive witnesses was not available in earlier builds.

Continuent 2.0.1 is the first generally available release of Continuent Tungsten 2.0, which offers major improvements to Continuent’s industry-leading database-as-a-service offering. Continuent Tungsten 2.0.1 contains all the improvements incorporated in Version 1.5.4, and the fixes and new features included within Tungsten Replicator 2.2.0, as well as the following features:

• Cluster Management
  • An improved manager that simplifies recovery of your cluster.
  • New tools to make provisioning and recovery of replication issues.
  • Improved witness host and decision engine to provide better quorum for preventing split-brain and prevent multiple live masters.
  • SSL-based encryption and authentication for cluster management through all command-line tools.

• Connector
  • SSL support enables SSL and non-SSL clients, and SSL and non-SSL connectivity between the connector and database servers.
  • Support for setting the maximum latency for slaves when redirecting queries.

• Installation and Deployment
  • Improved tpm installation tool that eases deployment and configuration of all clusters, including multi-master and multi-site/multi-master.
  • INI file based installation through tpm that enables easier installation, including through Puppet and other script-based solutions.

• Core Replication
  • Includes all Tungsten Replicator 2.2.0 features, including low-impact, low-latency replication, advanced filtering
• Supports MySQL (5.0, 5.1, 5.5, 5.6), MariaDB (5.5) and Percona Server (5.5).
• Supports replication to and from MySQL and Oracle, and Oracle to Oracle.
• Data loading to Vertica and InfiniDB, and real-time publishing to MongoDB.
• SSL-based encryption for exchanging replication data.

Behavior Changes

The following changes have been made to Continuent Tungsten and may affect existing scripts and integration tools. Any scripts or environment which make use of these tools should check and update for the new configuration:

• When using the xtrabackup method for performing backups, the default is to use the xtrabackup-full operation to perform a full backup.
  
  Issues: TUC-1327

• The default load balancer used for load-balancing connections within the Connector has been updated to use the RO_RELAXED_QoS balancer. This takes account of the HighWater mark when redirecting queries and compares the applied sequence number rather than relying only on the latency.
  
  Issues: TUC-1589

• Current strategy for preventing split-brain by using a witness host is not workable for many customers. The witness host configuration and checks have been changed to prevent these problems.
  
  Issues: TUC-1650

• Failover could be rolled back because of a failure to release a Virtual IP. The failure has been updated to trigger a warning, not a rollback of failover.
  
  Issues: TUC-1666

• An ‘UnknownHostException’ would cause a failover. The behavior has been updated to result in a suspect DB server.
  
  Issues: TUC-1667

• A new type of witness host has been added. A new active witness supports a manager-only based installation. The active witness is able to take part in decisions about failure in the event of datasource and/or network connectivity issues.

  As a result, the following changes apply for all witness host selection and installation:

  • Witnesses must be on the same network subnet as the existing managers.
  • Dataservices must have at least three managers to provide status check during failure.
  • Active witnesses can be created; these install only the manager on target hosts to act witnesses to check network connectivity to the configured dataserver and connectors configured within the service.

  Issues: TUC-1854

  For more information, see Section 2.1, "Host Types".

• Failover does not occur if the manager is not running, on the master host, before the time that the database server is stopped.
  
  Issues: TUC-1900

• Read-only MySQL slaves no longer work.
  
  Issues: TUC-1903

Improvements, new features and functionality

• Installation and Deployment

  • tpm has been updated to support configuration of the maximum applied latency for the connector using either the --connector-max-slave-latency [225] or --connector-max-applied-latency [225] options.

  Issues: TUC-733
• Installer should provide a way to setup RO_RELAXED (read-only with no SQL checking) connectors.
  
  **Issues:** TUC-954

• Post-installation notes do not specify hosts that can run `cctrl`.

  **Issues:** TUC-1118

• Create a `tpm cook` command that masks the tungsten-cookbook script

  **Issues:** TUC-1182

• The `tpm` validation has been updated to provided warnings when the `sync_binglog` and `innodb_flush_log_at_trx_commit` MySQL options are set incorrectly.

  **Issues:** TUC-1656

• A new `tpm` command has been added to list different connector connection commands/syntax.

  **Issues:** TUC-1661

• Add default path to security files, to facilitate their retrieval.

  **Issues:** TUC-1676

• Support a `--dataservice-witnesses` value of "none"

  **Issues:** TUC-1715

• The `tpm` command should not be accessible on installed data sources.

  **Issues:** TUC-1717

• Allow `tpm` configuration that is compatible with puppet/chef/etc

  **Issues:** TUC-1735

• Auto-generated properties line should go at the top of the files.

  **Issues:** TUC-1739

• Add `tpm` switch for `rrIncludeMaster` router properties.

  **Issues:** TUC-1744

• During installation, the `security.access_file.location` property should be changed to `security.rmi.jmxremote.access_file.location`.

  **Issues:** TUC-1805

• Split the cross machine checks out of `MySQLPermissionsCheck`.

  **Issues:** TUC-1838

• The installation of Multi-Site Multi-Master deployments has been simplified.

  **Issues:** TUC-1923

For more information, see Section 2.5, "Deploying a Multisite/Multimaster Topology".

• **Command-line Tools**

• A completion script for command-line completion within `bash` has been added to the installation. The file is located in `tools/.tpm.complete` within the installation directory.

  **Issues:** TUC-1591

• Write scripts to coordinate backups across an entire cluster.

  **Issues:** TUC-1641

• `CCTRL` should not report that recover is an expert command
Issues: TUC-1839

- An option, `-a, --authenticate` has been added to the `tpasswd` utility to validate an existing password entry.

Issues: TUC-1916

- Cookbook Utility
  - Tungsten cookbook should run manager|replicator|connector dump before collecting logs.
    
    Issues: TUC-1660
  - Cookbook has been updated to support both active and passive witnesses.
    
    Issues: TUC-1942
  - Cookbook has been updated to allow backups from masters to be used.
    
    Issues: TUC-1943

- Backup and Restore
  - The `datasource_backup.sh` script has been updated to limit running only on the `COORDINATOR` and to find a non-MASTER datasource.
    
    Issues: TUC-1684

- MySQL Replication
  - Add support for MySQL 5.6
    
    Issues: TUC-1624

- Tungsten Connector
  - Support for MySQL 4.0 passwords within the connector has been included. This provides support for both old MySQL versions and older versions of the MySQL protocol used by some libraries and clients.
    
    Issues: TUC-784
  - Connector must forbid zero `keepAliveTimeout`.
    
    Issues: TUC-1714
  - In SOR deployments only, Connector logs show relay data service being added twice.
    
    Issues: TUC-1720
  - Change default `delayBeforeOfflineIfNoManager` router property to 30s and constrain it to max 60s in the code.
    
    Issues: TUC-1752
  - Router Manager connection timeout should be a property.
    
    Issues: TUC-1754
  - Add client IP and port when logging connector message.
    
    Issues: TUC-1810
  - Make `tungsten cluster status` more sql-like and reduce the number of informations displayed.
    
    Issues: TUC-1814
  - Connector client side SSL support for MySQL
    
    Issues: TUC-1825

- Tungsten Manager
  - `cctrl` should show if a given data source is secured.
    
    Issues: TUC-1816
• The `datasource hostname recover` command should not invoke the expert warning.
  
  **Issues**: TUC-1840

**Manager API**

• Smarter enabling of the Manager API
  
  **Issues**: TUC-1621

• Support has been added to specify the addresses for the Manager API to listen on.
  
  **Issues**: TUC-1643

• The Manager API has been updated with a method to list all the available dataservices.
  
  **Issues**: TUC-1674

• Add DataServiceState and DataSource into the payload when applicable
  
  **Issues**: TUC-1701

• Add classes to the Ruby libraries that handle API calls
  
  **Issues**: TUC-1707

• Add an API call that prints the manager live properties
  
  **Issues**: TUC-1713

**Platform Specific Deployments**

• Add Java wrapper support for FreeBSD.
  
  **Issues**: TUC-1632

• Commit FreeBSD fixes to Java sockets and port binding.
  
  **Issues**: TUC-1633

**Documentation**

• Document among the prerequisites that Tungsten installers do not support `mysqld_multi`.
  
  **Issues**: TUC-1679

**Other Issues**

• Write a `tpm test` wrapper for the cookbook testing scripts.
  
  **Issues**: TUC-1396

• Document the process of sending emails based on specific log4j messages
  
  **Issues**: TUC-1500

• The `check_tungsten.sh` script has been updated to check and restart enterprise load balancers that use the `xinetd` service.
  
  **Issues**: TUC-1573

• Expand zabbix monitoring to match nagios checks.
  
  **Issues**: TUC-1638

• Turn SET NAMES log message into DEBUG.
  
  **Issues**: TUC-1644

• Remove old/extra/redundant configuration files.
  
  **Issues**: TUC-1721

• Backport critical 1.5.4 manager changes to 2.0.1
Issues: TUC-1855

Bug Fixes

• Installation and Deployment

  • Tungsten can't install if the 'mysql' client is not in the path.
    
    **Issues:** TUC-999

  • An extra `-l` flag when running `sudo` command would be added to the configuration.
    
    **Issues:** TUC-1025

  • Installer will not easily work when installing SOR data services one host at a time.
    
    **Issues:** TUC-1036

  • The `tpm` did not verify that the permissions for the `tungsten` DB user allow for cross-database host access.
    
    **Issues:** TUC-1146

  • Specifying a Symbolic link for the Connector/J creates a circular reference.
    
    **Issues:** TUC-1567

  • Replication of `DATETIME` values with a Daylight Savings Time (DST) would replicate incorrect values. Installation of a replication service where there are different timezones for the Java environment and the MySQL environment may cause incorrect replication.
    
    **Issues:** 542, TUC-1593

  • The replicator service would not be imported into the cluster directory - causes subsequent failures in switch and other operations.
    
    **Issues:** TUC-1594

  • `tpm` would fail to skip the `GlobalHostAddressesCheck` when performing a `tpm configure` followed by `tpm validate`.
    
    **Issues:** TUC-1599

  • `tpm` does not recognize datasources when they start with capital letter.
    
    **Issues:** TUC-1655

  • Installation of multiple replicator with `tpm` fails.
    
    **Issues:** TUC-1680

  • The check for Java version fails when OpenJDK does not say "java".
    
    **Issues:** TUC-1681

  • The installer did not make sure that witness servers are in the same network as the cluster.
    
    **Issues:** TUC-1705

  • `tpm` does not install if there is a Tungsten Replicator installer already running.
    
    **Issues:** TUC-1712

  • Errors during installation of composite dataservice.
    
    **Issues:** TUC-1726

  • The `tpm` command returns an `ssh` error when attempting to install a composite data service.
    
    **Issues:** TUC-1727

  • Running `tpm` with no arguments raises an error.
    
    **Issues:** TUC-1788

  • Installation fails with Ruby 1.9.
**Issues:** TUC-1800

- **tpm** will not throw an error if the user gives the connectorj-path as the path to a symlink instead of a real file.

**Issues:** TUC-1815

- **tpm** does not check dependencies of security options.

**Issues:** TUC-1818

- When checking process limits during installation, the check would fail the installation process instead of providing a warning.

**Issues:** TUC-1822

- During **tpm** validation wrongly complains about a witness not being in the same subnet.

**Issues:** TUC-1848

- During installation, **tpm** could install SSL support for the connector even though the MySQL server has not been configured for SSL connectivity.

**Issues:** TUC-1909

- Running **tpm update** would cause the master replicator to become a slave during the update when the master had changed from the configuration applied using `--dataservice-master-host [237].`

**Issues:** TUC-1921

- **tpm** could allow meaningless specifications of active witnesses.

**Issues:** TUC-1941

- **tpm** has been updated to provide the correct link to the documentation for further information.

**Issues:** TUC-1947

- Performing **tpm reset** would remove all the files within the `cluster-home/conf` directories, instead of only the files for services **tpm** was aware of.

**Issues:** TUC-1949

- **tpm** would require the `--active-witnesses [231]` or `--enable-active-witnesses [231]` option, when other witness types are available for configuration.

**Issues:** TUC-1951

- **tpm** would check the same witness subnet when using active witnesses, which do not need to be installed on the same subnet.

**Issues:** TUC-1953

- A **tpm update** operation would not recognize active witnesses properly.

**Issues:** TUC-1975

- A **tpm uninstall** operation would complain about missing databases in connector tests.

**Issues:** TUC-1978

- **tpm** would not remove the `connector.ro.properties` file if the configuration is updated to not have `--application-readonly-port [220].`

**Issues:** TUC-1981

- **tpm** would enable installation when MariaDB 10.0 was installed, even though this is not a supported configuration.

**Issues:** TUC-1987

- The method used to compare whether hosts were on the same subnet would fail to identify hosts correctly.

**Issues:** TUC-1995

**Command-line Tools**
• Running `cctrl` on a host which only had the connector server would not report a useful error. This has now been updated to show a warning message.

  *Issues: TUC-1642*

• The `check_tungsten` command had different command line arguments from `check_tungsten.sh`.

  *Issues: TUC-1675*

• Nagios check scripts not picking up shunned datasources

  *Issues: TUC-1689*

• `cctrl` could output the status of a host with a `null` value in place of the correct hostname.

  *Issues: TUC-1893*

• Using the `recover datasource` command within a composite service would fail, even though `datasource recover` would work.

  *Issues: TUC-1912*

• The `check_tungsten_latency --perslave-perfdata [155]` option would not include information for relay hosts.

  *Issues: TUC-1915*

• A large error message could be found included within the status block of `ls` output within `cctrl`. The error message information has been redirected to the error log.

  *Issues: TUC-1931*

• Performing `switch` operations within a composite service using active witnesses could raise an error and fail.

  *Issues: TUC-1946*

• `cctrl` would be unable to create a composite datasource after dropping it.

  *Issues: TUC-1956*

• Backwards compatibility for the `recover using` has been incorporated.

  *Issues: TUC-1971*

• **Cookbook Utility**

  • The tungsten-cookbook tests fails and does not print current status.

    *Issues: TUC-1623*

  • The `tungsten-cookbook` uses `resolveip` instead of standard name resolution tools.

    *Issues: TUC-1646*

  • The `tungsten-cookbook` tool sometimes misunderstands the result of composite recovery.

    *Issues: TUC-1662*

  • Cookbook gets warnings when used with a MySQL 5.6 client.

    *Issues: TUC-1673*

  • The cookbook does not wait for a database server to be offline properly.

    *Issues: TUC-1685*

  • `tungsten-cookbook` does not check the status of the relay server after a composite recovery.

    *Issues: TUC-1695*

  • `tungsten-cookbook` does not check all the components of a datasource when testing a server.

    *Issues: TUC-1696*
• **tungsten-cookbook** does not collect the configuration files under cluster-home.
  
  **Issues:** TUC-1697

• Cookbook should not specify witness hosts in default configuration files etc.
  
  **Issues:** TUC-1734

• Tungsten cookbook fails the replicator test.
  
  **Issues:** TUC-1827

• Using a backup that has been copied across servers within cookbook could overwrite or replace existing backup files, which would then make the backup file appear as older than it should be, making it unavailable in restore operations.
  
  **Issues:** TUC-1936

**Backup and Restore**

• The mysqldump backup option cannot restore if slow_query_log was on during the backup process.
  
  **Issues:** TUC-586

• Using xtrabackup during restore fails if MySQL is running as user ‘anything-but-mysql’ and without root access.
  
  **Issues:** TUC-1005

• When using mysqldump restore, the operation failed to disable slow and general logging before applying the restore.
  
  **Issues:** TUC-1330

• Backup fails when using the xtrabackup-full agent.
  
  **Issues:** TUC-1612

• Recovery hangs with composite data service.
  
  **Issues:** TUC-1657

• Performing a restore with xtrabackup fails.
  
  **Issues:** TUC-1672

• The datasource backup operation could fail due to a Ruby error.
  
  **Issues:** TUC-1686

• Restore with xtrabackup fails.
  
  **Issues:** TUC-1716

• Issues when recovering a failed physical dataservice.
  
  **Issues:** TUC-1793

• Backup with xtrabackup fails if datadir is not defined in my.cnf.
  
  **Issues:** TUC-1821

• When using xtrabackup restore fails.
  
  **Issues:** TUC-1846

• After a restore, datasource is welcomed and put online, but never gets to the online state.
  
  **Issues:** TUC-1861

• A restore that occurs immediately after a recover from dataserver failure always fails.
  
  **Issues:** TUC-1870

• Master datasource backup generates superficial failure message but succeeds anyway.
Restoration of a full backup would fail due to the inclusion of the `xtrabackup_incremental_basedir` directory.

Backup using `xtrabackup` 1.6.5 would fail.

When using the backup files copied from another server, the replicator could mistakenly use the wrong backup files when performing a restore.

**Core Replicator**

- Master failure causes partial commits on the slave with single channel parallel apply.
- Slave applier can fail to log error when DBMS fails due to exception in cleanup.
- Replication would fail on slave due to null characters created when inserting `___SERVICE___` comments.
- `LOAD (LOCAL) DATA INFILE` would fail if the request starts with white spaces.
- Datasource with a replicator in `GOING-ONLINE:RESTORING` shows up with a replicator `state=UNKNOWN`.
- An insecure slave can replicate from secure master.
- Replicator does not drop client connection to master and reconnect within the same time frame as in previous releases.

**Filters**

- Primary key filter should be able to renew its internal connection after some timeout.

**Tungsten Connector**

- TSR Session not updated when the database name changes (with `sessionid` set to `DATABASE`)
- Router gateway can prevent manager startup if the connector is started before the manager
- The Tungsten `show processlist` command would throw NPE errors.
- Selective read/write splitting (SQL-Based routing) has been updated to ensure that it is backwards compatible with previous read/write splitting configurations.
- Router must go into fail-safe mode if it loses connectivity to a manager during a critical command.
• Use of the `SET NAMES` command were not forwarded to attached read-only connections.
  
  **Issues**: TUC-1569

• When using **haproxy** through a connector connection, the initial query would be rejected.
  
  **Issues**: TUC-1581

• When the `dataservices.properties` file is empty, the connector would hang. The operation has now been updated to exit with an exception if the file cannot be found.
  
  **Issues**: TUC-1586

• When in a SOR deployment, the Connector will never return connection requests with `qos=RO_RELAXED` and affinity set to `relay node only Site`.
  
  **Issues**: TUC-1620

• Affinity not honored when using direct connections.
  
  **Issues**: TUC-1628

• Connector queries for `SHOW SLAVE STATUS` return incorrect slave latency of 0 intermittently.
  
  **Issues**: TUC-1645

• The Tungsten Connector does not know it’s PID following upgrade to JSW 3.5.17.
  
  **Issues**: TUC-1665

• An attempt to load a driver listener class can cause the connector to hang, at startup.
  
  **Issues**: TUC-1669

• Read connections allocated by connector get 'stale' and are closed by MySQL server due to `wait_timeout` - causes app 'transparency' issues.
  
  **Issues**: TUC-1671

• Broken connections returned to the c3p0 pool - further use of these will show errors.
  
  **Issues**: TUC-1683

• Router disconnects from a manager in the middle of a `switch` command - writes continue to offline master.
  
  **Issues**: TUC-1692

• Connector sessionId passed in database name not retained
  
  **Issues**: TUC-1704

• When using `USE DB` within a connector after the database had previously been dropped would be incorrectly ignored.
  
  **Issues**: TUC-1718

• The connector `tungsten flush privileges` command causes a temporary outage (denies new connection requests).
  
  **Issues**: TUC-1730

• Database context not changed to the correct database when `qos=DATABASE` is in use.
  
  **Issues**: TUC-1779

• Connector should require a valid manager to operate even when in maintenance mode.
  
  **Issues**: TUC-1781

• Connector allows connections to an offline/on-hold composite dataservice.
  
  **Issues**: TUC-1787
• Router notifications are being sent to routers via GCS. This is unnecessary since a manager only updates routers that are connected to it.
  
  **Issues:** TUC-1790

• Pass through not handling correctly multiple results in 1.5.4.
  
  **Issues:** TUC-1792

• SmartScale will fail to create a database and use immediately.
  
  **Issues:** TUC-1836

• The connector could hang during installation test.
  
  **Issues:** TUC-1847

• Under certain circumstances, SSL-configuration for the Connector would be unable to start properly.
  
  **Issues:** TUC-1869

  For more information, see Section 3.3.4, "Configuring Connector SSL".

• Specify where to load security properties from in the connector.
  
  **Issues:** TUC-1872

• A **SET NAMES** operation would not survive a **switch** or **failover** operation.
  
  **Issues:** TUC-1879

• The connector command within **cctrl** has been disabled unless the connector and manager are installed on the same host.

To support the removed functionality, the following changes to the connector command have been made:

• The **wildcard** can be used for connectors within the **router** command within **cctrl**. For example, **router * online** will place all available connectors online.

• The built-in command-line completion provides the names of the connectors in addition to the **wildcard** character for the **router** command.
  
  **Issues:** TUC-1918

• Using cursors within stored procedures through the connector would cause a hang in the connector service.
  
  **Issues:** TUC-1950

• The connector would hang when working in a cluster with active witnesses.
  
  **Issues:** TUC-1954

• When specifying the affinity within a connection, the **maxAppliedLatency** configuration would be ignored.
  
  **Issues:** TUC-1960

• The connector would check for changes to the **user.map** frequently, causing lag on high-load servers. The configuration has been updated to allow checking only every 10s.
  
  **Issues:** TUC-1972

• Passing the **qos** option within a database name would not work when smart scale was enabled.
  
  **Issues:** TUC-1982

• Tungsten Manager

  • The **datasource restore** command may fail when using **xtrabackup** if the file ownership for the backup files is wrong.
    
    **Issues:** TUC-1226

  • Dataservice has different "composite" status depending on how its status is called.
Release Notes

Issues: TUC-1614

- The `switch` command does not validate command line correctly.
  
  **Issues:** TUC-1618

- Composite recovery would fail because a replicator that was previously a master tries to re-apply a transaction that it had previously committed.
  
  **Issues:** TUC-1634

- `cctrl` would let you shun the master datasource.
  
  **Issues:** TUC-1637

- During a failover, the master could be left in read-only mode.
  
  **Issues:** TUC-1648

- On occasion, the manager would fail to restart after being hung.
  
  **Issues:** TUC-1649

- The ping command in `cctrl` wrongly identifies witness server as unreachable.
  
  **Issues:** TUC-1652

- The failure of primary data source could go unhandled due to a manager restart.
  
  **Issues:** TUC-1659

- The manager reports composite recovery completion although the operation has failed.
  
  **Issues:** TUC-1663

- A transient error can cause a confused state.
  
  **Issues:** TUC-1678

- Composite recovery could fail, but the manager says it was complete.
  
  **Issues:** TUC-1694

- The internal Call to `OpenReplicatorManager.status()` during transition from online to offline results in a `NullPointerException`.
  
  **Issues:** TUC-1708

- Relay does not fail over when the database server is stopped.
  
  **Issues:** TUC-1711

- The `cctrl` would raise an error when running a backup from a master.
  
  **Issues:** TUC-1789

- Tungsten manager may report false host failures due to a temporary problem with name resolution.
  
  **Issues:** TUC-1797

- `cctrl` could report a manager as `ONLINE` even though the datasource would in fact be `OFFLINE`.
  
  **Issues:** TUC-1804

- The manager would not see a secured replicator.
  
  **Issues:** TUC-1806

- Slave replicators never come online after a switch when using secure thl.
  
  **Issues:** TUC-1807

**cctrl** complains of missing security file when security is not enabled.
Issues: TUC-1808

• Switch in relay site fails and takes offline all nodes.

Issues: TUC-1809

• A switch in the relay site sets the relay to replicate from itself.

Issues: TUC-1811

• In a composite deployment, a switch in the primary site is not propagated to the relay.

Issues: TUC-1813

• `cctrl` exposes security passwords unnecessarily.

Issues: TUC-1817

• The master datasource is not available following the `failover` command.

Issues: TUC-1841

• The manager does not support a non-standard replicator RMI port.

Issues: TUC-1842

• In a multi-site deployment, automatic failover does not happen in maintenance mode, due to replicator issues.

Issues: TUC-1845

• During the recovery of a composite dataservice, the restore of a shunned master could fail because the previous and current roles did not match.

Issues: TUC-1857

• A stopped dataserver would not be detected if cluster was in maintenance mode when it was stopped.

Issues: TUC-1860

• Manager attempts to get status of remote replicator from the local service - causes a failure to catch up from a relay.

Issues: TUC-1864

• A `switch` operation could fail in single site deployment.

Issues: TUC-1867

• In a configuration with a relay of a composite site, if all active data datasources are unavailable, a `switch` operation would raise invalid exception messages.

Issues: TUC-1875

• `recover using` fails in the simplest case for 2.0.1.

Issues: TUC-1876

• Manager fails safe even if it is in the quorum set and primary partition.

Issues: TUC-1878

• Single command `recover` does not work - does not find datasources to recover even if they exist.

Issues: TUC-1881

• Failover causes old master node name to disappear from `cctrl ls` command.

Issues: TUC-1894

• ClusterManagementHandler can read/write datasources directly from the local disk - can cause cluster configuration information corruption.
• Stopping managers does not cause membership validation rules to kick in. This can lead to an invalid group.
  
  *Issues*: TUC-1901

• The manager rules could fail to fence a composite datasource for which all managers in the service are unreachable.
  
  *Issues*: TUC-1902

• `recover using` in a master service could convert one of the datasources into a relay instead of a slave.
  
  *Issues*: TUC-1907

• `CREATE COMPOSITE DATASOURCE` could result in an exception if the master datasource site was used.
  
  *Issues*: TUC-1911

• The manager would throw a false alarm if the `trep_commit_seqno` table was empty. This was due to the manager being started before the replicator had created the required table.
  
  *Issues*: TUC-1917

• Composite recovery within a cloud deployment could fail.
  
  *Issues*: TUC-1922

• Errors could be raised when using the `set master` and `recover using` commands within `cctrl`.
  
  *Issues*: TUC-1930

• Composite recovery could fail in a site with multiple masters.
  
  *Issues*: TUC-1932

• A failed master within a dataservice would cause the datasource names to disappear.
  
  *Issues*: TUC-1933

• Running `switch` command after performing recovery could fail within a multi-site deployment.
  
  *Issues*: TUC-1934

• Performing a `switch` operation when there are active witness could cause an error message indicating a fault, when in fact the operation completed successfully.
  
  *Issues*: TUC-1935

• After performing a `switch` operation, a slave could report to the previous, not active, relay.
  
  *Issues*: TUC-1939

• Running operations on active witness datasources would raise `nullPointerException` errors.
  
  *Issues*: TUC-1944, TUC-1945

• Errors would be reported in the log when deserializing configuration information between the manager and connector.
  
  *Issues*: TUC-1963

• Automatic failover would fail to run if an active witness was the coordinator for the dataservice.
  
  *Issues*: TUC-1964

• Connectors would disappear after restarting the coordinator.
  
  *Issues*: TUC-1966

• The coordinator would attempt to check database server liveness if a manager on a witness host goes away.
  

• Composite recovery using a streaming backup results in a site with multiple masters.
Issues: TUC-1992
• Installing a composite dataservice would create two master services.

Issues: TUC-1996

• Manager API
  • API call for a single server does not report replicator status.
    Issues: TUC-1615
  • API "promote" command does not operate in a composite dataservice.
    Issues: TUC-1617
  • Some indispensable commands missing from manager API.
    Issues: TUC-1654
  • Manager API does not answer to /manager/status/svc_name without Accept header
    Issues: TUC-1690
  • The Manager API lets you shun a master.
    Issues: TUC-1706
  • The call to 'policy' API fails in composite dataservice.
    Issues: TUC-1725

• Platform Specific Deployments
  • Windows service registration scripts won't work.
    Issues: TUC-1636
  • FreeBSD: Replicator hangs when going offline. Can cause switch to hang/abort.
    Issues: TUC-1668

• Documentation
  • Document the process for changing the replication username and password.
    Issues: TUC-638
    For more information, see ???.
  • Documentation has been added for deploying Continuent Tungsten with INI files
    Issues: TUC-1888
    For more information, see Section 7.9.4, "tpm INI File Configuration".
  • Documentation on the different tpm commands has been added to the documentation.
    Issues: TUC-1890
    For more information, see Section 7.9.5, "tpm Commands".
  • Documentation for the new tools designed to ease usability with Continuent Tungsten have been added.
    Issues: TUC-1891, TUC-1892
    For more information, see Section 7.14, "The tungsten_health_check Script", Section 7.11, "The tungsten_monitor Script", Section 7.10, "The tungsten_provision_slave Script", Section 7.12, "The tungsten_read_master_events Script", Section 7.13, "The tungsten_set_position Script".

• Other Issues
  • The shared libraries used by Continuent Tungsten have now been centralized in the cluster-home directory.
Issues: TUC-1310
• Some build warnings in Java 1.6 become errors in Java 1.7.

Issues: TUC-1731
• The test_connection_routing_and_isolation.rb test never selects the correct master.

Issues: TUC-1780
• During testing, a test that stops and restarts the replicator fails because a replicator that is actually running shows up, subsequently, as stopped.

Issues: TUC-1895
• The wrapper for the service was not honoring the configured wait period during a restart, which could cause a hang or failure when the service was restarted.

Issues: TUC-1910, TUC-1913

Continuent Tungsten 2.0.1 Includes the following changes from Tungsten Replicator 2.2.0

Tungsten Replicator 2.2.0 is a bug fix and feature release that contains a number of key improvements to the installation and management of the replicator:
• tpm is now the default installation and deployment tool; use of tungsten-installer, configure, configure-service, and update are deprecated.
• tpm incorporates support for both INI file and staging directory deployments. See Section 7.9.4, "tpm INI File Configuration".
• Deployments are possible using standard Linux RPM and PKG deployments. See Section 2.3.2, "Using the RPM and DEB package files".
• tpm has been improved to handle heterogeneous deployments more easily.
• New command-line tools have been added to make recovery easier during a failure. See Section 7.10, "The tungsten_provision_slave Script", Section 7.12, "The tungsten_read_master_events Script", Section 7.13, "The tungsten_set_position Script".
• Improvements to the core replicator, including identification and recovery from failure.
• New multi_trepctl tool for monitoring multiple hosts/services.

Behavior Changes
The following changes have been made to Tungsten Replicator and may affect existing scripts and integration tools. Any scripts or environment which make use of these tools should check and update for the new configuration:
• The following commands to trepctl have been deprecated and will be removed in a future release:
  • trepctl start has been replaced with trepctl load
  • trepctl stop has been replaced with trepctl unload
  • trepctl shutdown has been deprecated; use Section 2.12, "Starting and Stopping Continuent Tungsten" to stop the replicator.

Issues: 672
For more information, see Section 7.8.3.8, "trepctl load Command", Section 7.8.3.21, "trepctl unload Command", Section 2.12, "Starting and Stopping Continuent Tungsten".
• The tpm command has been updated to be the default method for installing deployments using the cookbook. To use the old tungsten-installer command, set the USE_OLD_INSTALLER environment variable.

Issues: 691

Improvements, new features and functionality
• Installation and Deployment

  • For heterogeneous deployments, three new options have been added to `tpm`:

    • `--enable-heterogenous-master [232]`

      This option applies a range of settings, including `--mysql-use-bytes-for-string=false`, `--java-file-encoding=UTF8 [235]`, `--mysql-enable-enumtostring=true`, and `--mysql-enable-settostring=true [240]`. This option also enables the `colnames` and `pkey` filters.

    • `--enable-heterogenous-slave [232]`

      This option disables parallel replication for hosts that do not support it, and sets the `--java-file-encoding=UTF8 [235]` option.

    • `--enable-heterogenous-service [232]`

      Enables the `--enable-heterogenous-master [232]` and `--enable-heterogenous-slave [232]` for masters and slaves respectively.

  Issues: 692

  For more information, see Installing MongoDB Replication in [Tungsten Replicator 2.2 Manual], Installing Vertica Replication in [Tungsten Replicator 2.2 Manual].

• Command-line Tools

  • A new command-line tool, `tungsten_set_position`, has been created. This enables the position of either a master or slave to be set with respect to reading local or remote events. This provides easier control over during the recovery of a slave or master in the event of a failure.

  Issues: 684

  For more information, see Section 7.13, "The `tungsten_set_position` Script", Section 4.7, "Managing Transaction Failures".

  • A new command-line tool, `tungsten_provision_slave`, has been created. This allows for an automated backup of an existing host and restore of that data to a new host. The script can be used to provision new slaves based on existing slave configurations, or to recover a slave that has failed.

  Issues: 689

  For more information, see Section 7.10, "The `tungsten_provision_slave` Script", Section 4.7, "Managing Transaction Failures".

  • A new command-line tool, `tungsten_read_master_events`, has been created. This enables events from the MySQL binary log to be viewed based on the THL event ID.

  Issues: 694

  For more information, see Section 7.12, "The `tungsten_read_master_events` Script", Section 4.7, "Managing Transaction Failures".

  • The `trepctl properties` command has been updated to support a `--values` option that outputs only the values for filtered properties.

  Issues: 719

  For more information, see Section 7.8.3.12, "trepctl properties Command".

• Oracle Replication

  • The `ddlscan` tool and the `ddl-mysql-oracle.vm` template have been modified to support custom included templates on a per table basis.

  The tool has also been updated to support additional paths for searching for velocity templates using the `--path` option.

  Issues: 723
• The block commit process has been updated to support different configurations. Two new parameters have been added, which affect the block commit size, and enable transactions to be committed to a slave in blocks either based on the number of events, or the time interval since the last commit occurred.
  • \texttt{--repl-svc-applier-block-commit-size} sets the number of events that will trigger a block commit. The default is 10.
  • \texttt{--repl-svc-applier-block-commit-interval} sets the time interval between block commits. The default is 0 (disabled).

\textit{Issues: 677, 699}

For more information, see Section 9.1, "Block Commit".

• Filters
  • The \texttt{dropcolumn} JavaScript filter has been added. The filter enables individual columns to be removed from the THL so that personal identification information (PII) can be removed on a slave.

\textit{Issues: 716}

For more information, see Section 8.5.2.6, "\texttt{dropcolumn.js Filter}".

Bug Fixes

• Installation and Deployment
  • When performing a Vertica deployment, \texttt{tpm} would fail to create the correct configuration parameters. In addition, error messages and warnings would be generated that did not apply to Vertica installations. \texttt{tpm} has been updated to simplify the Vertica installation process.

\textit{Issues: 688, 781}

For more information, see Installing Vertica Replication in [Tungsten Replicator 2.2 Manual].

• When configuring a single host to support a parallel, multi-channel deployment, \texttt{tpm} would report that this operation was not supported. \texttt{tpm} has now been updated to support single host parallel apply configurations.

\textit{Issues: 737}

• Configuring an installation with a preferred path for MySQL deployments using the \texttt{--preferred-path} \texttt{[242]} option would not set the \texttt{PATH} variable correctly, this would lead to the tools from an incorrect directory being used when performing backup or restore operations. \texttt{tpm} has been updated to correctly set the environment during execution.

\textit{Issues: 752}

• Command-line Tools
  • When using the \texttt{-sql} option to the \texttt{thl}, additional metadata and options would be displayed. The tool has now been updated to only output the corresponding SQL.

\textit{Issues: 264}

• \texttt{DATETIME} values could be displayed incorrectly in the THL when using the \texttt{thl} tool to show log contents.

\textit{Issues: 676}

• An incorrect RMI port could be used within a deployment if a non-standard RMI port was specified during installation, affecting the operation of \texttt{trepctl}. The precedence for selecting the RMI port to use has been updated to use the \texttt{-port}, the system property, and then service properties for the selected service and/or \texttt{trepctl} executable.

\textit{Issues: 695}

• Backup and Restore
  • During installation, \texttt{tpm} would fail to check the version for Percona XtraBackup when working with built-in InnoDB support in MySQL. The check has now been updated and validation will fail if XtraBackup 2.1 or later is used with a MySQL 5.1 and built-in InnoDB support.

\textit{Issues: 671}
• When using `xtrabackup` during a restore operation, the restore would fail. The problem was due to a difference in the interface for XtraBackup 2.1.6.

  Issues: 778

**Oracle Replication**

• When performing an Oracle deployment, `tpm` would apply incorrect parameters and filters and check MySQL specific environment information. The following changes have been made:
  - The `colnames` filter is no longer added to Oracle master (extractor) deployments.
  - Incorrect schema value would be defined for the replicator schema.

   The check for `mysqldump` is still performed on an Oracle master host; use `--preferred-path [242]` to set a valid location, or disable the `MySqlDumpCheck` validation check.

  Issues: 685

**Core Replicator**

• `DECIMAL` values could be extracted from the MySQL binary log incorrectly when using statement based logging.

  Issues: 650

• A null pointer exception could be raised by the master, which would lead to the slave failing to connect to the master correctly. The slave will now retry the connection.

  Issues: 698

• A slave replicator could fail when synchronizing the THL if the master goes offline. This was due to network interrupts during a failure not being recognised properly.

  Issues: 714

• In certain circumstances, a replicator could apply transactions that had been generated by itself. This could happen during a failover, leading to events written to the THL, but without the `trep_commit_seqno` table having been updated. To fix this problem, consistency checks on the THL contents are now performed during startup. In addition, all replicators now write their currently assigned role to a file within the configuration directory of the running replication service, called `static-servicename.role`.

   When the replicator goes online, a `static-servicename.role` file is examined. If the current role identified in that file was a master, and the current role of the replicator is a slave, then the THL consistency checks are enabled. These check the following situations:
   - If the `trep_commit_seqno` is out of sync with the contents of the THL provided that the last THL record exists and matches the source-id of the transaction.
   - If the current log position is different to the THL position, and assuming that THL position exists, then an error will be raised and the replicator will go offline. This behavior can be overridden by using the `trepctl online -force` command.

   Once the checks have been completed, the new role for the replicator is updated in the `static-servicename.role` file.

   **Important**

   The `static-servicename.role` file must be deleted, or the THL files must be deleted, when restoring a backup. This is to ensure that the correct current log position is identified.

   Issues: 735

• An `UnsupportedEncodingException` error could occur when extracting statement based replication events if the MySQL character set did not match a valid Java character set used by the replicator.

  Issues: 743

• When using Row-based replication, replicating into a table on the slave that did not exist, a Null-Pointer Exception would be raised. The replicator now correctly raises an SQL error indicating that the table does not exist.

  Issues: 747
• During a master failure under load, the number of transactions making it to the slave before the master replicator fails.


Issues: 753

• Upgrading a replicator and changing the hostname could cause the replicator to skip events in the THL. This was due to the way in which the source-id of events in the slave replicator checks the information compared to the remote THL read from the master. This particularly affect standalone replicators. The fix adds a new property, replicator.repositionOnSourceIdChange. This is a boolean value, and specifies whether the replicator should try to reposition to the correct location in the THL when the source ID has been modified.


Issues: 754

• Running trepctl reset on a service deployed in an multi-master (all master) configuration would not correctly remove the schema from the database.


Issues: 758

• Replication of temporary tables with the same name, but within different sessions would cause a conflict in the slave.


Issues: 772

• Filters

• The PrimaryKeyFilter would not renew connections to the master to determine the primary key information. When replication had been running for a long time, the active connection would be dropped, but never renewed. The filter has been updated to re-connect on failure.


Issues: 670

For more information, see Section 8.4.17, “PrimaryKey Filter”.
Appendix C. Prerequisites

Before you install Continuent Tungsten, there are a number of setup and prerequisite installation and configuration steps that must have taken place before any installation can continue. Section C.1, "Staging Host Configuration" and Section C.2, "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 C.3, "MySQL Database Setup", and will need to be performed on each appropriate host.

C.1. 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 Continuent Tungsten™ software, and to install, transfer, and initiate the Continuent Tungsten™ 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 Continuent Tungsten™ 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 C.1, “Tungsten Deployment”.

Figure C.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 Continuent Tungsten™. The recommended choice for MySQL installations is to create a new user, tungsten. You will need to create this user on each host in the cluster. You can create the new user using adduser:

   ```shell
   sudo adduser tungsten
   ```

   You can add the user to the mysql group adding the command-line option:
Prerequisites

2. Login as the tungsten user:

```shell
su - tungsten
```

3. Create an SSH key file, but do not configure a password:

```
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:
```
```
```
This creates both a public and private keyfile; the public keyfile will be shared with the hosts in the cluster to allow hosts to connect to each other.

4. Within the staging server, profiles for the different cluster configurations are stored within a single directory. You can simplify the management of these different services by configuring a specific directory where these configurations will be stored. To set the directory, specify the directory within the `CONTINUENT_PROFILES` environment variable, adding this variable to your shell startup script (`.bashrc`, for example) within your staging server.

```
shell> mkdir -p /opt/continuent/software/conf
shell> mkdir -p /opt/continuent/software/replicator.conf
shell> export CONTINUENT_PROFILES=/opt/continuent/software/conf
shell> export REPLICATOR_PROFILES=/opt/continuent/software/replicator.conf
```

We now have a staging server setup, an SSH keypair for our login information, and are ready to start setting up each host within the cluster.

C.2. 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 authorisation for the user on each host
- Configuring the directories and install locations
- Installing necessary software and tools
- Configuring `sudo` access to enable the configured user to perform administration commands

**Important**

The operations in the following sections must be performed on each host within your cluster. Failure to perform each step may prevent the installation and deployment of Tungsten cluster.

C.2.1. 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`:
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:

```plaintext
# Tungsten user
tungsten    -    nofile    65535

# Database user
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:

```plaintext
# Tungsten user
tungsten    -    nproc    8096

# Database user
mysql       -    nproc    8096
```

You must logout and log back in again for the `ulimit` changes to take effect.

**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`:

```plaintext
session required pam_limits.so
```

Make sure that Apparmor, if configured, has been enabled to support access to the `/tmp` directory for the MySQL processes. For example, add the following to the MySQL configuration file (usually `/etc/apparmor.d/local/usr.sbin.mysqld`):

```
/tmp/** rwk
```

---

### C.2.2. 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
  127.0.0.1 host1
  ```

- Each host in the cluster must be able to resolve the address for all the other hosts in the cluster. To prevent errors within the DNS system causing timeouts or bad resolution, all hosts in the cluster, in addition to the witness host, should be added to `/etc/hosts`:

  ```
  127.0.0.1 localhost
  127.0.0.1 host1
  192.168.1.60 host2
  192.168.1.61 host3
  192.168.1.62 host4
  192.168.1.63 host5
  ```

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.
Prerequisites

- The IP address of each host within the cluster must resolve to the same IP address on each node. For example, if host1 resolves to 192.168.0.69 on host1, the same IP address must be returned when looking up host1 on the host host2.

To double check this, you should perform the following tests:

1. Confirm the hostname:
   
   ```shell
   # Confirm the hostname:
   shell> hostname -n
   ```

   **Warning**
   The hostname cannot contain underscores.

2. Confirm the IP address:
   
   ```shell
   # 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
   # Confirm that the hostnames of the other hosts in the cluster resolve correctly to a valid IP address:
   shell> nslookup host1
   shell> ping host1
   ```

   If the host does not resolve, either ensure that the hosts are added to the DNS service, or explicitly add the information to the /etc/hosts file.

   **Warning**
   If using /etc/hosts then you must ensure that the information is correct and consistent on each host, and double check using the above method that the IP address resolves correctly for every host in the cluster.

### Witness Hosts

Continuent Tungsten includes support for verifying the network status using a *witness* host.

- **Active Witness Hosts**: operate as standalone managers, and therefore require the same rights and requirements as a standard Continuent Tungsten host.
- **Passive Witness Hosts**: can be any stable network device. The passive witness host will be contacted using `ping` in the event of a network failure to confirm that network services are operational and that the problem is with an individual node.

### C.2.2.1. 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>7800-7805</td>
<td>Manager Remote Method Invocation (RMI)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>9997</td>
<td>Manager Remote Method Invocation (RMI)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10000-10001</td>
<td>Replication connection listener port</td>
</tr>
<tr>
<td>Connector Service</td>
<td>Connector Host</td>
<td>Manager Hosts</td>
<td>11999</td>
<td>Tungsten manager</td>
</tr>
<tr>
<td>Connector Service</td>
<td></td>
<td></td>
<td>13306</td>
<td>Database connectivity</td>
</tr>
<tr>
<td>Client Application</td>
<td>Client</td>
<td>Connector</td>
<td>3306</td>
<td>Database connectivity for client</td>
</tr>
</tbody>
</table>

For composite clusters, communication between each cluster within the composite configuration can be limited to the following ports:

<table>
<thead>
<tr>
<th>Component</th>
<th>Port</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database service</td>
<td>99997</td>
<td>Manager Remote Method Invocation (RMI)</td>
</tr>
<tr>
<td></td>
<td>2112</td>
<td>THL replication</td>
</tr>
</tbody>
</table>
### Prerequisites

<table>
<thead>
<tr>
<th>Component</th>
<th>Port</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tungsten Manager</td>
<td>11999-12000</td>
<td>Tungsten Manager</td>
</tr>
<tr>
<td>Client Application</td>
<td>13306</td>
<td>MySQL port for Connectivity</td>
</tr>
<tr>
<td>Manager Hosts</td>
<td>7</td>
<td>Communication between managers within composite clusters</td>
</tr>
</tbody>
</table>

For Multisite, Multimaster (MSMM) clusters that communicate through replication, the communication between sites can be limited to the following ports:

<table>
<thead>
<tr>
<th>Component</th>
<th>Port</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tungsten</td>
<td>2114</td>
<td>THL replication</td>
</tr>
<tr>
<td>Tungsten</td>
<td>10002-10003</td>
<td>Replication connection listener ports</td>
</tr>
<tr>
<td>Client Application</td>
<td>13306</td>
<td>MySQL port for Connectivity</td>
</tr>
<tr>
<td>Manager Hosts</td>
<td>7</td>
<td>Communication between managers within multi-site, multi-master clusters</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 --state NEW -j ACCEPT
```

### C.2.2.2. 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.

### C.2.2.3. Host Availability Checks

The manager checks the availability of other hosts, for example to determine whether the host is still up, rather than just an individual service on that host. These checks must be able to be performed by one of the two available methods, using the `ping` or `echo` protocols.

Without these checks, it is possible for the availability of hosts to be falsely determined. These checks are performed using one of two protocols:

- **default** — the default method is to use the TCP/IP echo protocol on port 7. The port must be available on the source and destination host, not blocked by a system or network firewall.

  This may require enabling the protocol and the supported daemon on all nodes within the cluster, for example by enabling the `echo` protocol within `xinetd` or similar.

- **ping** — the system `ping` (ICMP) command. This must be enabled, and, if necessary, not restricted by firewall or other rules that would prevent communication between hosts or clusters.

One of these two solutions must be used.

The configuration of which service to use depends on the setting of the `policy.liveness.hostPingMethods` property, which can be configured after the service has been installed.

In addition the `policy.liveness.hostPingTimeout` can be configured to set the timeout value when the host liveness is checked. The default is to wait for 2 seconds. Because the liveness is checked both within the local cluster and to remote clusters in multi-site deployments, the
timeout may need to be increased to take account of the network latency. Using a setting above 5 seconds may lead to problems due to the timing of other operations and services.

C.2.3. Directory Locations and Configuration

On each host within the cluster you must pick, and configure, a number of directories to be used by Continuent Tungsten™, 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
  
  Continuent Tungsten™ needs to be installed in a specific directory. The recommended solution is to use /opt/continuent. This information will be required when you configure the cluster service.

  The directory should be created, and the owner and permissions set for the configured user:

  ```
  shell> sudo mkdir /opt/continuent
  shell> sudo chown tungsten /opt/continuent
  shell> sudo chmod 700 /opt/continuent
  ```

- Home Directory
  
  The home directory of the tungsten user must be writable by that user.

C.2.4. Configure Software

Continuent Tungsten™ 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>Ruby</td>
<td>1.8.7, 1.9.3, or 2.0.0 or higher[^1]</td>
<td>JRuby is not supported</td>
</tr>
<tr>
<td>Ruby OpenSSL Module</td>
<td>-</td>
<td>Checking using <code>ruby -ropenssl -e 'p &quot;works&quot;'</code></td>
</tr>
<tr>
<td>GNU tar</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Java Runtime Environment</td>
<td>Java SE 6 or 7 (or compatible)</td>
<td></td>
</tr>
<tr>
<td>MySQL Connector/J</td>
<td>5.1.18 or later</td>
<td>Download from Connector/J</td>
</tr>
</tbody>
</table>

[^1]: Ruby 1.9.1 and 1.9.2 are not supported; these releases remove the execute bit during installation.

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

  ```
  shell> java -version
  Java version "1.6.0_31"
  Java(TM) SE Runtime Environment (build 1.6.0_31-b04)
  Java HotSpot(TM) 64-Bit Server VM (build 20.6-b01, mixed mode)
  ```

  On certain environments, a separate tool such as alternatives 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 4.11, “Performing Database or OS Maintenance” procedures.
Prerequisites

It also recommended to install ntp or a similar time synchronization tool so that each host in the cluster has the same physical time.

C.2.5. 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/tar, /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/tar, /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
```

C.3. 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.

Note

Native MySQL replication should not be running when you install Continuent Tungsten™. The replication service will be completely handled by Continuent Tungsten™, and the normal replication, management and monitoring techniques will not provide you with the information you need.

C.3.1. 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 must be able to read the MySQL configuration file (for installation) and the binary logs. Either the tungsten user should be a member of the appropriate group (i.e. mysql), or the permissions altered accordingly.

Important

Parsing of mysqld_multi configuration files is not currently supported. To use a mysqld_multi installation, copy the relevant portion of the configuration file to a separate file to be used during installation.
To setup your MySQL servers, you need to do the following:

• Configure your `my.cnf` Settings. The following changes should be made to the `[mysqld]` section of your `my.cnf` file:

  • By default, MySQL is configured only to listen on the localhost address (127.0.0.1). The `bind-address` parameter should be checked to ensure that it is either set to a valid value, or commented to allow listening on all available network interfaces:

    ```
    [mysqld]
    bind-address = 127.0.0.1
    ```

  • Specify the server id

    Each server must have a unique server id:

    ```
    server-id = 1
    ```

  • (Optional) Reconfigure the default MySQL TCP/IP port

    Change the listening port to 13306. The Tungsten Connector will listen on the normal port 3306 for MySQL connections and send them to the database using port 13306:

    ```
    port = 13306
    ```

    If you are not using Tungsten Connector, the setting can remain at the default of 3306.

  • 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).

    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

    Continuent Tungsten 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 Continuent Tungsten.

    ```
    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 Continuent Tungsten™, 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 Continuent Tungsten™ 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 Continuent Tungsten™ 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 Continuent Tungsten™ 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 4.11, "Performing Database or OS Maintenance".

Optional configuration changes that can be made to your MySQL configuration:

• InnoDB Flush Method

```
inoddb_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 3.2, "Deploying Parallel Replication".

• Increase InnoDB log file size

The default InnoDB log file size is 5MB. This should be increased to a larger file size, due to a known issue with xtrabackup during backup and restore operations.

To change the file size, read the corresponding information in the MySQL manual for configuring the file size information. See MySQL 5.1, MySQL 5.5, MySQL 5.6, MySQL 5.7.

• 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
```

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.

### C.3.2. MySQL User Configuration

#### Tungsten User Login

The `tungsten` user connects to the MySQL database and applies the data from the replication stream from other datasources in the dataservice. The user must therefore be able execute any SQL statement on the server, including grants for other users. The user must have the following privileges in addition to privileges for creating, updating and deleting DDL and data within the database:

- **SUPER** privilege is required so that the user can perform all administrative operations including setting global variables.
- **GRANT OPTION** privilege is required so that users and grants can be updated.

To create a user with suitable privileges:

```
mysql> CREATE USER tungsten@'%' IDENTIFIED BY 'password';
mysql> GRANT ALL ON *.* TO tungsten@'%' WITH GRANT OPTION;
```

The connection will be made from the host to the local MySQL server. You may also need to create an explicit entry for this connection. For example, on the host `host1`, create the user with an explicit host reference:

```
mysql> CREATE USER tungsten@'host1' IDENTIFIED BY 'password';
mysql> GRANT ALL ON *.* TO tungsten@'host1' WITH GRANT OPTION;
```

The above commands enable logins from any host using the user name/password combination. If you want to limit the configuration to only include the hosts within your cluster you must create and grant individual user/host combinations:

```
mysql> CREATE USER tungsten@'client1' IDENTIFIED BY 'password';
mysql> GRANT ALL ON *.* TO tungsten@'client1' WITH GRANT OPTION;
```

**Note**

If you later change the cluster configuration and add more hosts, you will need to update this configuration with each new host in the cluster.

#### MySQL Application Login

Tungsten Connector requires a user that can be used as the application user to connect to the MySQL server. The login will allow connections to the MySQL databases servers to be used in a consistent fashion across different hosts within the cluster. You must configure this user with access to your database, and then use it as the ‘application’ user in your cluster configuration.

```
mysql> CREATE USER app_user@'%' IDENTIFIED BY 'password!';
mysql> GRANT ALL ON *.* TO app_user@'%';
mysql> REVOKE SUPER ON *.* FROM app_user@'%';
```

Additional application user logins can be configured by using the `user.map` file within your Continuent Tungsten™ configuration.

As noted above, the creation of explicit host-specific user entries may be required.
C.4. Oracle Database Setup

C.4.1. 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><code>/home/oracle/app/oracle/product/11.2.0/dbhome_2</code></td>
<td>The home directory of the Oracle installation.</td>
</tr>
<tr>
<td>LD_LIBRARY_PATH</td>
<td><code>$ORACLE_HOME/lib</code></td>
<td>The library directory of the Oracle installation.</td>
</tr>
<tr>
<td>ORACLE_SID</td>
<td><code>orcl</code></td>
<td>Oracle System ID for this installation.</td>
</tr>
<tr>
<td>JAVA_HOME</td>
<td></td>
<td>The home of the Java installation.</td>
</tr>
<tr>
<td>PATH</td>
<td><code>$ORACLE_HOME/bin:$JAVA_HOME/bin</code></td>
<td>Must include the Oracle and Java binary directories.</td>
</tr>
<tr>
<td>CLASSPATH</td>
<td><code>$ORACLE_HOME/ucp/lib/ucp.jar:$ORACLE_HOME/jdbc/lib/ojdbc6.jar:$CLASSPATH</code></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.

C.5. PostgreSQL Database Setup
Appendix D. Terminology Reference

Continuent Tungsten 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 Continuent Tungsten.

D.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 the `logDir` parameter of the THL component. A sample directory is shown below:

```bash
Total 710504
-rw-r--r-- 1 tungsten tungsten 0 May  2 10:48 disklog.lck
-rw-rw-r-- 1 tungsten tungsten 100042908 Jun  4 10:10 thl.data.0000000013
-rw-rw-r-- 1 tungsten tungsten 101025311 Jun  4 11:41 thl.data.0000000014
-rw-rw-r-- 1 tungsten tungsten 100441159 Jun  4 11:43 thl.data.0000000015
-rw-rw-r-- 1 tungsten tungsten 100898492 Jun  4 11:44 thl.data.0000000016
-rw-rw-r-- 1 tungsten tungsten 100305613 Jun  4 11:44 thl.data.0000000017
-rw-rw-r-- 1 tungsten tungsten 10035516 Jun  4 11:44 thl.data.0000000018
-rw-rw-r-- 1 tungsten tungsten 101690969 Jun  4 11:45 thl.data.0000000019
-rw-rw-r-- 1 tungsten tungsten 30046641 Jun  5 21:50 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 `logFileSize` configuration parameter. The log files are automatically managed by Tungsten Replicator, with old files automatically removed according to the retention policy set by the `logFileRetention` configuration parameter. The files can be manually purged or moved. See Section E.1.4.1, "Purging THL Log Information".

The THL can be viewed and managed by using the `thl` command. For more information, see Section 7.7, "The thl Command".

D.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:

```bash
SEQ# = 0 / FRAG# = 0 (last frag)
- TIME = 2013-03-21 18:47:39.0
- EPOCH# = 0
- EVENTID = mysql-bin.000010:0000000000000439;0
- SOURCEID = host1
- METADATA = [mysql_server_id=10;dbms_type=mysql;is_metadata=true;service=dsone;»
  shard=tungsten_firstcluster;heartbeat=MASTER_ONLINE]
- TYPE = com.continuent.tungsten.replicator.event.ReplDBMSEvent
```
The sample above shows the information for the SQL executed on a MySQL server. The `event_id` 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 D.1, "THL Event Format".

Table D.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>Length of the full record information, including this field</td>
<td></td>
</tr>
<tr>
<td>record_type</td>
<td>Byte</td>
<td>1 byte</td>
<td>Event record type identifier</td>
<td></td>
</tr>
<tr>
<td>header_length</td>
<td>Unsigned int</td>
<td>4 bytes</td>
<td>Length of the header information</td>
<td></td>
</tr>
<tr>
<td>seqno [359]</td>
<td>Unsigned long</td>
<td>8 bytes</td>
<td>Log sequence number, a sequential value given to each log entry</td>
<td></td>
</tr>
<tr>
<td>fragno [359]</td>
<td>Unsigned short</td>
<td>2 bytes</td>
<td>Event fragment number. An event can consist of multiple fragments of SQL or row log data</td>
<td></td>
</tr>
<tr>
<td>last_frag</td>
<td>Byte</td>
<td>1 byte</td>
<td>Indicates whether the fragment is the last fragment in the sequence</td>
<td></td>
</tr>
<tr>
<td>epoch_number</td>
<td>Unsigned long</td>
<td>8 bytes</td>
<td>Event epoch number. Used to identify log sections within the master THL</td>
<td></td>
</tr>
<tr>
<td>source_id [360]</td>
<td>UTF-8 String</td>
<td>Variable (null terminated)</td>
<td>Event source ID, the hostname or identity of the dataserver that generated the event</td>
<td></td>
</tr>
<tr>
<td>event_id [360]</td>
<td>UTF-8 String</td>
<td>Variable (null terminated)</td>
<td>Event ID; in MySQL, for example, the binlog filename and position that contained the original event</td>
<td></td>
</tr>
<tr>
<td>shard_id [361]</td>
<td>UTF-8 String</td>
<td>Variable (null terminated)</td>
<td>Shard ID to which the event belongs</td>
<td></td>
</tr>
<tr>
<td>tstamp [360]</td>
<td>Unsigned long</td>
<td>8 bytes</td>
<td>Time of the commit that triggered the event</td>
<td></td>
</tr>
<tr>
<td>data_length</td>
<td>Unsigned int</td>
<td>4 bytes</td>
<td>Length of the included event data</td>
<td></td>
</tr>
<tr>
<td>event</td>
<td>Binary</td>
<td>Variable</td>
<td>Serialized Java object containing the SQL or ROW data</td>
<td></td>
</tr>
<tr>
<td>crc_method [361]</td>
<td>Byte</td>
<td>1 byte</td>
<td>Method used to compute the CRC for the event.</td>
<td></td>
</tr>
<tr>
<td>crc</td>
<td>Unsigned int</td>
<td>4 bytes</td>
<td>CRC of the event record (not including the CRC value)</td>
<td></td>
</tr>
</tbody>
</table>

- **SEQUENCE** and **FRAGMENT**

Individual events within the log are identified by a sequential `SEQUENCE` 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#**
The `EPOCH` value is used as a check to ensure that the logs on the slave and the master match. The `EPOCH` is stored in the THL, and a new `EPOCH` is generated each time a master goes online. The `EPOCH` value is then written and stored in the THL alongside each individual event. The `EPOCH` acts as an additional check, beyond the sequence number, to validate the information between the slave and the master. The `EPOCH` 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` acts as a check so that when the master rejoins as the slave, the `EPOCH` 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` prevents the slave from accepting events that happen to match only the sequence number, but not the corresponding `EPOCH`.

Each time a Tungsten Replicator master goes online, the `EPOCH` number is incremented. When the slave connects, it requests the `SEQUENCE` and `EPOCH`, and the master confirms that the requested `SEQUENCE` has the requested `EPOCH`. If not, the request is rejected and the slave gets a validation error:

```plaintext
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 4.7, “Managing Transaction Failures”.

- `SOURCEID`

  The `SOURCEID` is a string identifying the source of the event stored in the THL. Typically it is the hostname or host identifier.

- `EVENTID`

  The `EVENTID` is a string identifying the source of the event information in the log. Within a MySQL instance, the `EVENTID` 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`

  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` value as stored in the THL is used to compute latency information when reading and applying data on a slave.

- `METADATA`

  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`

  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`
Part of the `EVENT` binary payload, the `OPTIONS` 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` block for each `SQL` statement stored in the event.

- **SCHEMA**
  Part of the `EVENT` structure, the `SCHEMA` provides the database or schema name in which the statement or row data was applied.

- **SHARDID**
  When using parallel apply, provides the generated shard ID for the event when it is applied by the parallel applier thread.

- **SQL**
  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:

```sql
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]
- 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`, `TABLE` and individual `ROW` data. For each `ROW`, there may be one or more `COL` (column) and identifying `KEY` event to identify the row on which the action is to be performed.

The same statement when recorded in `ROW` 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(1) = INSERT INTO user VALUES (null, 'Charles', now()) /* ___SERVICE___ = [firstrep] */
- ACTION = INSERT
- SCHEMA = test
- TABLE = user
- ROW = 0
  - COL[1: ] = 2
  - COL[2: ] = Charles
```

### D.2. Generated Field Reference

When using any of the tools within Continuent Tungsten 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.

**D.2.1. Terminology: Fields**

- **accessFailures**

**D.2.2. Terminology: Fields**

- **active**
D.2.3. Terminology: Fields  

**activeSeqno**

D.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. For example, when extracting from MySQL, the output from `trepctl` shows the MySQL binary log file and position within the log where the transaction was extracted:

```
shell> trepctl status
Processing status command...
NAME VALUE
---- -----
appliedLastEventId : mysql-bin.000064:0000000002757461;0
...```

D.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.

D.2.6. Terminology: Fields  

**appliedLatency**

The applied latency is the latency between the commit time of the source event and the time the last committed transaction reached the end of the corresponding pipeline within the replicator.

Within a 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.

Inreplicators 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.

D.2.7. Terminology: Fields  

**applier.class**

Classname of the current applier engine

D.2.8. Terminology: Fields  

**applier.name**

Name of the current applier engine

D.2.9. Terminology: Fields  

**applyTime**

D.2.10. Terminology: Fields  

**autoRecoveryEnabled**

Indicates whether autorecovery has been enabled by setting the `--auto-recovery-max-attempts` flag. The field indicates the value as either `true` or `false` accordingly.

D.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 `trepsvc.log` file.

The counter is reset when the replicator determines that the replicator has successfully gone online after an autorecovery.

D.2.12. Terminology: Fields  

**averageBlockSize**
D.2.13. Terminology: Fields blockCommitRowCount


D.2.15. Terminology: Fields channel

D.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.

D.2.17. Terminology: Fields clusterName

The name of the cluster. This information is different from the service name and is used to identify the cluster, rather than the individual service information being output.

D.2.18. Terminology: Fields commits

D.2.19. Terminology: Fields committedMinSeqno

D.2.20. Terminology: Fields criticalPartition

D.2.21. Terminology: Fields currentBlockSize

D.2.22. Terminology: Fields currentEventId

Event ID of the transaction currently being processed

D.2.23. Terminology: Fields currentLastEventId

D.2.24. Terminology: Fields currentLastFragno

D.2.25. Terminology: Fields currentLastSeqno


The current time on the host, in milliseconds since the epoch. This information can be used to confirm that the time on different hosts is within a suitable limit. Internally, this information is used to record the time when transactions are applied, and may therefore the appliedLatency figure.

D.2.27. Terminology: Fields dataServerHost

D.2.28. Terminology: Fields discardCount

D.2.29. Terminology: Fields doChecksum
D.2.30. Terminology: Fields estimatedOfflineInterval

D.2.31. Terminology: Fields eventCount

D.2.32. Terminology: Fields extensions

D.2.33. Terminology: Fields extractTime

D.2.34. Terminology: Fields extractor.class

D.2.35. Terminology: Fields extractor.name

D.2.36. Terminology: Fields filter.#.class

D.2.37. Terminology: Fields filter.#.name

D.2.38. Terminology: Fields filterTime

D.2.39. Terminology: Fields flushIntervalMillis

D.2.40. Terminology: Fields fsyncOnFlush

D.2.41. Terminology: Fields headSeqno

D.2.42. Terminology: Fields intervalGuard

D.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 9.1, "Block Commit".

D.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 9.1, "Block Commit".

D.2.45. Terminology: Fields latestEpochNumber

D.2.46. Terminology: Fields logConnectionTimeout
D.2.47. Terminology: Fields logDir

D.2.48. Terminology: Fields logFileRetainMillis

D.2.49. Terminology: Fields logFileSize

D.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/
```

D.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 : thls://host1:2112/
```

D.2.52. Terminology: Fields maxChannel

D.2.53. Terminology: Fields maxDelayInterval

D.2.54. Terminology: Fields maxOfflineInterval

D.2.55. Terminology: Fields maxSize

D.2.56. Terminology: Fields maximumStoredSeqNo

The maximum transaction ID that has been stored locally on the machine in the THL. Because Tungsten Replicator operates in stages, it is sometimes important to compare the sequence and latency between information being ready from the source into the THL, and then from the THL into the database. You can compare this value to the `appliedLastSeqno`, which indicates the last sequence committed to the database. The information is provided at a resolution of milliseconds.

```
maximumStoredSeqNo : 25
```

D.2.57. Terminology: Fields minimumStoredSeqNo

The minimum transaction ID stored locally in the THL on the host:

```
minimumStoredSeqNo : 0
```

The figure should match the lowest transaction ID as output by the `thl index` command. On a busy host, or one where the THL information has been purged, the figure will show the corresponding transaction ID as stored in the THL.

D.2.58. Terminology: Fields name

D.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:
D.2.60. Terminology: Fields `otherTime`

D.2.61. Terminology: Fields `pendingError`


D.2.63. Terminology: Fields `pendingErrorEventId`

D.2.64. Terminology: Fields `pendingErrorSeqno`

The sequence number where the current error was identified

D.2.65. Terminology: Fields `pendingExceptionMessage`

The current error message that caused the current replicator offline

D.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.

D.2.67. Terminology: Fields `processedMinSeqno`

D.2.68. Terminology: Fields `queues`

D.2.69. Terminology: Fields `readOnly`

D.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.

D.2.71. Terminology: Fields `resourcePrecedence`

D.2.72. Terminology: Fields `rmiPort`

D.2.73. Terminology: Fields `role`

The current role of the host in the corresponding service specification. Primary roles are `master` and `slave`. 
D.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.

D.2.75. Terminology: Fields serializationCount

D.2.76. Terminology: Fields serialized

D.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.

D.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.

D.2.79. Terminology: Fields shard_id

D.2.80. Terminology: Fields simpleServiceName

A simplified version of the serviceName.

D.2.81. Terminology: Fields siteName

D.2.82. Terminology: Fields sourceId

D.2.83. Terminology: Fields stage

D.2.84. Terminology: Fields started

D.2.85. Terminology: Fields state

D.2.86. Terminology: Fields stopRequested

D.2.87. Terminology: Fields store.$

D.2.88. Terminology: Fields storeClass

D.2.89. Terminology: Fields syncInterval

D.2.90. Terminology: Fields taskCount
D.2.91. Terminology: Fields taskId

D.2.92. Terminology: Fields timeInStateSeconds

D.2.93. Terminology: Fields timeoutMillis

D.2.94. Terminology: Fields totalAssignments

D.2.95. Terminology: Fields transitioningTo

D.2.96. Terminology: Fields uptimeSeconds

D.2.97. Terminology: Fields version
Appendix E. Files, Directories, and Environment

E.1. The Continuent Tungsten Install Directory

Any Continuent Tungsten™ 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 E.1, “Continuent Tungsten Directory Structure”.

```
shell> ls -al /opt/continuent
 total 40
 drwxr-xr-x 9 tungsten root 4096 Mar 21 18:47 .
 drwxr-xr-x 3 root     root     4096 Mar 21 18:00 ..
 drwxrwxr-x 2 tungsten tungsten 4096 Mar 21 18:44 backups
 drwxrwxr-x 2 tungsten tungsten 4096 Mar 21 18:47 conf
 drwxrwxr-x 3 tungsten tungsten 4096 Mar 21 18:44 relay
 drwxrwxr-x 4 tungsten tungsten 4096 Mar 21 18:47 releases
 drwxrwxr-x 2 tungsten tungsten 4096 Mar 21 18:47 service_logs
 drwxrwxr-x 2 tungsten tungsten 4096 Mar 21 18:47 share
 drwxrwxr-x 3 tungsten tungsten 4096 Mar 21 18:44 thl
 lrwxrwxrwx 1 tungsten tungsten   62 Mar 21 18:47 tungsten -> /opt/continuent/releases/continuent-tungsten-2.0.5-3_pid31409
```

The directories shown in the table are relative to the installation directory, the recommended location is /opt/continuent. For example, the THL files would be located in /opt/continuent/thl.

Table E.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.

E.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 Continuent Tungsten™. The backup data and metadata for each backup will be stored in this directory.

An example of the directory content is shown below:

```
shell> ls -al /opt/continuent/backups/
 total 130788
 drwxr-xr-x 2 tungsten tungsten 4096 Apr  4 16:09 .
 drwxr-xr-x 3 tungsten tungsten 4096 Apr  4 11:51 ..
-rw-r--r-- 1 tungsten tungsten   71 Apr  4 16:09 storage.index
-rw-r--r-- 1 tungsten tungsten 133907646 Apr  4 16:09 store-0000000001-mysqldump_2013-04-04_16-08_42.sql.gz
-rw-r--r-- 1 tungsten tungsten   317 Apr  4 16:09 store-0000000001.properties
```

The storage.index contains the backup file index information. The actual backup data is stored in the GZipped file. The properties of the backup file, including the tool used to create the backup, and the checksum information, are located 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.

E.1.1.1. Automatically Deleting Backup Files

The Tungsten Replicator will automatically remove old backup files. This is controlled by the --repl-backup-retention 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.

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.

E.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
```

<table>
<thead>
<tr>
<th>File Name</th>
<th>Permissions</th>
<th>Owner</th>
<th>Size</th>
<th>Date/Time</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>drwxrwxr-x</td>
<td>tungsten</td>
<td>4096</td>
<td>Apr 16 13:57</td>
<td>.</td>
</tr>
<tr>
<td>..</td>
<td>drwxrwxr-x</td>
<td>tungsten</td>
<td>4096</td>
<td>Apr 16 13:54</td>
<td>..</td>
</tr>
<tr>
<td>storage.index</td>
<td>-rw-r--r--</td>
<td>tungsten</td>
<td>71</td>
<td>Apr 16 13:56</td>
<td>storage.index</td>
</tr>
<tr>
<td>store-0000000004-mysqldump-133246373891843527.sql</td>
<td>-rw-r--r--</td>
<td>tungsten</td>
<td>517170</td>
<td>Apr 15 18:02</td>
<td>store-0000000004-mysqldump-133246373891843527.sql</td>
</tr>
<tr>
<td>store-0000000004.properties</td>
<td>-rw-r--r--</td>
<td>tungsten</td>
<td>311</td>
<td>Apr 15 18:02</td>
<td>store-0000000004.properties</td>
</tr>
<tr>
<td>store-0000000005-mysqldump-2284057977980000458.sql</td>
<td>-rw-r--r--</td>
<td>tungsten</td>
<td>517170</td>
<td>Apr 15 18:06</td>
<td>store-0000000005-mysqldump-2284057977980000458.sql</td>
</tr>
<tr>
<td>store-0000000005.properties</td>
<td>-rw-r--r--</td>
<td>tungsten</td>
<td>310</td>
<td>Apr 15 18:06</td>
<td>store-0000000005.properties</td>
</tr>
<tr>
<td>store-0000000006-mysqldump-3081853249977885370.sql</td>
<td>-rw-r--r--</td>
<td>tungsten</td>
<td>781991444</td>
<td>Apr 16 13:57</td>
<td>store-0000000006-mysqldump-3081853249977885370.sql</td>
</tr>
<tr>
<td>store-0000000006.properties</td>
<td>-rw-r--r--</td>
<td>tungsten</td>
<td>314</td>
<td>Apr 16 13:57</td>
<td>store-0000000006.properties</td>
</tr>
</tbody>
</table>

To delete the backup files for index 4:

```
shell> rm /opt/continuent/backups/firstrep/store-0000000004*
```

See the information in Section E.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.

E.1.1.3. Copying Backup Files

The files created during any backup can be copied to another directory or system using any suitable means. Once the backup has been completed, the files will not be modified or updated and are therefore safe to be moved or actively copied to another location without fear of corruption of the backup information.

There are multiple files associated with each backup. The number of files will depend on the backup method that was used. All backups will use at least two files in the /opt/continuent/backups directory.

```
shell> cd /opt/continuent/backups
shell> scp store-[0-6][.-]* host3:$PWD/
```

- **store-0000000001-full_xtrabackup_2014-08-16_15-44_86** 100% 70 0.1KB/s 00:00
- **store-0000000001.properties** 100% 314 0.3KB/s 00:00

**Note**

Check the ownership of files if you have trouble transferring files or restoring the backup. They should be owned by the Tungsten system user to ensure proper operation.

If `xtrabackup-full` method was used, you must transfer the corresponding directory from /opt/continuent/backups/xtrabackup. In this example that would be `/opt/continuent/backups/xtrabackup/full_xtrabackup_2014-08-16_15-44_86`.

```
shell> cd /opt/continuent/backups/xtrabackup
shell> rsync --a --exclude-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 -ltr /opt/continuent/backups/xtrabackup/
```

<table>
<thead>
<tr>
<th>File Name</th>
<th>Permissions</th>
<th>Owner</th>
<th>Size</th>
<th>Date/Time</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>drwxr-xr-x</td>
<td>tungsten</td>
<td>4096</td>
<td>Oct 16 20:52</td>
<td>.</td>
</tr>
<tr>
<td>..</td>
<td>drwxr-xr-x</td>
<td>tungsten</td>
<td>4096</td>
<td>Oct 16 20:51</td>
<td>..</td>
</tr>
</tbody>
</table>

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_20-58_97`, it must be copied along with `full_xtrabackup_2014-10-18_20-55_38` and `full_xtrabackup_2014-10-18-20-55_1`.

E.1.1.4. Relocating Backup Storage

If the filesystem on which the main installation directory is running out of space and you need to increase the space available for backup files without interrupting the service, you can use symbolic links to relocate the backup information.

### Note

When using an NFS mount point when backing up with `xtrabackup`, the command must have the necessary access rights and permissions to change the ownership of files within the mounted directory. Failure to update the permissions and ownership will cause the `xtrabackup` command to fail. The following settings should be made on the directory:

- Ensure the `no_root_squash` option on the NFS export is not set.
- Change the group and owner of the mount point to the `tungsten` user and `mysql` group:
  ```
  shell> chown tungsten /mnt/backups
  shell> chgrp mysql  /mnt/backups
  ```

  Owner and group IDs on NFS directories must match across all the hosts using the NFS mount point. Inconsistencies in the owner and group IDs may lead to backup failures.

- Change the permissions to permit at least owner and group modifications:
  ```
  shell> chmod 770 /mnt/backups
  ```

- Mount the directory:
  ```
  shell> mount host1:/exports/backups /mnt/backups
  ```

The backup directory can be changed using two different methods:

- [Section E.1.1.4.1, ”Relocating Backup Storage using Symbolic Links”](#)
- [Section E.1.1.4.2, ”Relocating Backup Storage using Configuration Changes”](#)

E.1.1.4.1. Relocating Backup Storage using Symbolic Links

To relocate the backup directory using symbolic links:

1. Ensure that no active backup is taking place of the current host. Your service does not need to be offline to complete this operation.

2. Create a new directory, or attach a new filesystem and location on which the backups will be located. You can use a directory on another filesystem or connect to a SAN, NFS or other filesystem where the new directory will be located. For example:

   ```
   shell> mkdir /mnt/backupdata/continuent
   ```

3. Optional

   Copy the existing backup directory to the new directory location. For example:

   ```
   shell> rsync -r /opt/continuent/backups/* /mnt/backupdata/continuent/
   ```

4. Move the existing directory to a temporary location:

   ```
   shell> mv /opt/continuent/backups /opt/continuent/old-backups
   ```

5. Create a symbolic link from the new directory to the original directory location:

   ```
   shell> ln -s /mnt/backupdata/continuent /opt/continuent/backups
   ```

The backup directory has now been moved. If you want to verify that the new backup directory is working, you can optionally run a backup and ensure that the backup process completes correctly.

E.1.1.4.2. Relocating Backup Storage using Configuration Changes

To relocate the backup directory by reconfiguration:
1. Ensure that no active backup is taking place of the current host. Your service does not need to be offline to complete this operation.

2. Create a new directory, or attach a new filesystem and location on which the backups will be located. You can use a directory on another filesystem or connect to a SAN, NFS or other filesystem where the new directory will be located. For example:

   ```
   shell> mkdir /mnt/backupdata/continuent
   ```

3. Optional

   Copy the existing backup directory to the new directory location. For example:

   ```
   shell> rsync -r /opt/continuent/backups/* /mnt/backupdata/continuent/
   ```

4. Following the directions for `tpm update` to apply the `--backup-directory=/mnt/backupdata/continuent` 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.

**E.1.2. The releases Directory**

The `releases` directory contains a copy of each installed release. As new versions are installed and updated (through `tpm update`), a new directory is created with the corresponding version of the software.

For example, a number of releases are listed below:

```
shell> ll /opt/continuent/releases/
total 20
drwxr-xr-x  5 tungsten mysql 4096 May 23 16:19 ./
drwxr-xr-x  9 tungsten mysql 4096 May 23 16:19 ../
drwxr-xr-x 10 tungsten mysql 4096 May 23 16:19 continuum-tungsten-2.0.5-3_pid16184/
drwxr-xr-x 10 tungsten mysql 4096 May 23 16:19 continuum-tungsten-2.0.5-3_pid14577/
drwxr-xr-x 10 tungsten mysql 4096 May 23 16:19 continuum-tungsten-2.0.5-3_pid23747/
drwxr-xr-x 10 tungsten mysql 4096 May 23 16:19 continuum-tungsten-2.0.5-3_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 conf/
drwxr-xr-x  2 tungsten mysql 4096 May 23 16:19 relay/
drwxr-xr-x  2 tungsten mysql 4096 May 23 16:19 backups/
drwxr-xr-x  2 tungsten mysql 4096 May 23 16:19 service_logs/
```

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/firstrep/
total 2291984
```

Directories within the `release` directory that are no longer being used can be safely removed.

**E.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:

- `connector.log` — a link to the Tungsten Connector log.
- `tmsvc.log` — a link to the Continuent Tungsten manager log.
- `trepsvc.log` — a link to the Tungsten Replicator log.

**E.1.4. 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:
THL files are created on both the master and slaves within the cluster. THL data can be examined using the `thl` command. For more information, see Section 7.7, “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(756:852)
LogIndexEntry thl.data.0000000009(853:948)
LogIndexEntry thl.data.0000000010(949:1045)
LogIndexEntry thl.data.0000000011(1046:1141)
LogIndexEntry thl.data.0000000012(1142:1238)
LogIndexEntry thl.data.0000000013(1239:1334)
LogIndexEntry thl.data.0000000014(1335:1430)
LogIndexEntry thl.data.0000000015(1431:1526)
LogIndexEntry thl.data.0000000016(1527:1622)
LogIndexEntry thl.data.0000000017(1623:1718)
LogIndexEntry thl.data.0000000018(1719:1814)
```

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.

### E.1.4.1. Purging THL Log Information

**Warning**

Purging the THL can potentially remove information that has not yet been applied to the database. Please check and ensure that the THL data that you are purging has been applied to the database before continuing.

The THL files can be explicitly purged to recover disk space, but you should ensure that the currently applied sequence no to the database is not purged, and that additional hosts are not reading the THL information.

To purge the logs:
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: firstrep
serviceType: local
started: true
state: ONLINE
Finished services command...
```

2. Shun the datasource and switch your node into the offline state using `cctrl`:

```
shell> cctrl -expert
[LOGICAL:EXPERT] /alpha > datasource host1 shun
[LOGICAL:EXPERT] /alpha > replicator host1 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. Recover the host back into the cluster:

```
shell> cctrl -expert
[LOGICAL:EXPERT] /alpha > datasource host1 recover
```

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 7.7.3, "thl purge Command".

### E.1.4.2. 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 E.1.4.2.1, "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 E.1.4.2.2, "Relocating THL Storage using Configuration Changes".

#### E.1.4.2.1. 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. Shun the datasource and switch your node into the offline state using `cctrl`:

```
shell> cctrl -expert
[LOGICAL:EXPERT] /alpha > datasource host1 shun
[LOGICAL:EXPERT] /alpha > replicator host1 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:
4. Move the existing directory to a temporary location:

```
shell> mv /opt/continuent/thl/* /mnt/data/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. Recover the host back into the cluster:

```
shell> cctrl -expert

[LOGICAL:EXPERT] /alpha > datasource host1 recover
```

To change the THL location on a master:

1. Manually promote an existing slave to be the new master:

```
[LOGICAL] /alpha > switch to host2
SELECTED SLAVE: host2@alpha
PURGE REMAINING ACTIVE SESSIONS ON CURRENT MASTER 'host1@alpha'
PURGED A TOTAL OF 0 ACTIVE SESSIONS ON MASTER 'host1@alpha'
FLUSH TRANSACTIONS ON CURRENT MASTER 'host1@alpha'
PUT THE NEW MASTER 'host2@alpha' ONLINE
PUT THE PRIOR MASTER 'host1@alpha' ONLINE AS A SLAVE
RECONFIGURING SLAVE 'host3@alpha' TO POINT TO NEW MASTER 'host2@alpha'
SWITCH TO 'host2@alpha' WAS SUCCESSFUL
```

2. Update the THL location as provided in the previous sequence.

3. Switch the updated slave back to be the master:

```
[LOGICAL] /alpha > switch to host1
SELECTED SLAVE: host1@alpha
PURGE REMAINING ACTIVE SESSIONS ON CURRENT MASTER 'host2@alpha'
PURGED A TOTAL OF 0 ACTIVE SESSIONS ON MASTER 'host2@alpha'
FLUSH TRANSACTIONS ON CURRENT MASTER 'host2@alpha'
PUT THE NEW MASTER 'host1@alpha' ONLINE
PUT THE PRIOR MASTER 'host2@alpha' ONLINE AS A SLAVE
RECONFIGURING SLAVE 'host3@alpha' TO POINT TO NEW MASTER 'host1@alpha'
SWITCH TO 'host1@alpha' WAS SUCCESSFUL
```

E.1.4.2.2. 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. Shun the datasource and switch your node into the offline state using `cctrl`:

```
shell> cctrl -expert

[LOGICAL:EXPERT] /alpha > datasource host1 shun
[LOGICAL:EXPERT] /alpha > replicator host1 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. Recover the host back into the cluster:

```
shell> cctrl -expert

[LOGICAL:EXPERT] /alpha > datasource host1 recover
```

To change the THL location on a master:

1. Manually promote an existing slave to be the new master:

```
[LOGICAL] /alpha > switch to host2
```
2. Update the THL location as provided in the previous sequence.

3. Switch the updated slave back to be the master:

```plaintext
[LOGICAL] /alpha > switch to host1
SELECTED SLAVE: host1@alpha
PURGE REMAINING ACTIVE SESSIONS ON CURRENT MASTER 'host2@alpha'
PURGED A TOTAL OF 0 ACTIVE SESSIONS ON MASTER 'host2@alpha'
FLUSH TRANSACTIONS ON CURRENT MASTER 'host2@alpha'
PUT THE NEW MASTER 'host1@alpha' ONLINE
PUT THE PRIOR MASTER 'host2@alpha' ONLINE AS A SLAVE
RECONFIGURING SLAVE 'host3@alpha' TO POINT TO NEW MASTER 'host1@alpha'
SWITCH TO 'host1@alpha' WAS SUCCESSFUL
```

### E.1.4.3. 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 Continuent Tungsten. 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.

### E.1.5. The `tungsten` Directory

```plaintext
shell> ls -l /opt/continuent/tungsten/
```

<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 Continuent Tungsten installation.</td>
</tr>
<tr>
<td>cookbook</td>
<td>Cookbook installation and testing tools.</td>
</tr>
<tr>
<td>INSTALL</td>
<td>Text file describing the basic installation process for Continuent Tungsten</td>
</tr>
<tr>
<td>README.LICENSES</td>
<td>Software license information.</td>
</tr>
<tr>
<td>tools</td>
<td>Directory containing the tools for installing and configuring Continuent Tungsten.</td>
</tr>
<tr>
<td>tungsten-connector</td>
<td>Installed directory of the Tungsten Connector installation.</td>
</tr>
<tr>
<td>tungsten-manager</td>
<td>Installed directory of the Tungsten Manager installation.</td>
</tr>
<tr>
<td>tungsten-replicator</td>
<td>Installed directory of the Tungsten Replicator installation.</td>
</tr>
</tbody>
</table>

### E.2. 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 7.9.3, "tpm Staging Configuration".

- $REPLICATOR_PROFILES [377]

  When using `tpm` with Tungsten Replicator, `$REPLICATOR_PROFILES [377]` is used for storing the `deploy.cfg` file during configuration and installation. If `$REPLICATOR_PROFILES [377]` does not exist, then `$CONTINUENT_PROFILES [376]` if it exists. For more information, see Section 7.9.3, "tpm Staging Configuration".

- $CONTINUENT_ROOT [377]

  The `$CONTINUENT_ROOT [377]` variable is created by the `env.sh` file that is created when installing Continuent Tungsten. When defined, the variable will contain the installation directory of the corresponding Continuent Tungsten installation.

  On hosts where multiple installations have been created, the variable can be used to point to different installations.
Appendix F. Internals

Continuent Tungsten 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 Continuent Tungsten:

- **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 Continuent Tungsten”** — 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 Continuent Tungsten.

F.1. Extending Backup and Restore Behavior

The backup and restore system within Continuent Tungsten 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 Continuent Tungsten 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:

```
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 storage_file
    TU.cmd_result("echo 'my backup' > #{storage_file}")
  end
end
```
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` option should be set to `script`, and the directory location set using the `--repl-backup-script` 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` 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:
Internals

```
--property=replicator.backup.agent.script.options="arg1=val1&myarg=val2"
```

These are the custom parameters which are supplied to the script as the value of the \-options parameter when the script is called.

Once the backup script has been enabled within the configuration it can be used when performing a backup through the standard backup or restore interface:

For example, within cctrl:

```
[LOGICAL:EXPERT] /alpha > datasource host2 backup script
```

Note

Note that the name of the backup method is \texttt{script}, not the actual name of the script being used.

### F.2. Character Sets in Database and Continuent Tungsten

Character sets within the databases and within the configuration for Java and the wrappers for Continuent Tungsten 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 or MySQL) no explicit setting should be made. For heterogeneous deployments, the character set should be set explicitly.

When installing and using Continuent Tungsten, be aware of the following aspects when using character sets:

- When installing Continuent Tungsten, use the \texttt{--java-file-encoding [235]} to \texttt{tpm} to configure the character set.
- When using the \texttt{thl} command, the character set may need to be explicitly stated to view the content correctly:

  ```
  shell> thl list --charset utf8
  ```

For more information on setting character sets within your database, see your documentation for the database:

- MySQL
- Oracle

For more information on the character set names and support within Java, see:

- Java 6 SE
- Java 7 SE

### F.3. Replication of Date/Time Values

- Replicator processes default to UTC internally by setting the Java VM default time zone to UTC. This default can be changed by setting the replicator.time_zone property in the replicator services.propertiesx file but is not recommended other than for problem diagnosis or specialized testing.
- Replicas store a time zone on statements and row changes extracted from MySQL.
- Replicators use UTC as the session time zone when applying to MySQL replicas.
- Replicators similarly default to UTC when applying transactions to data warehouses like Hadoop, Vertica, or Amazon Redshift.
- The \texttt{thl} utility prints time-related data using the default GMT time zone. This can be altered using the \texttt{-timezone} option.

**Best Practices**

We recommend the following steps to ensure successful replication of time-related data:

- Standardize all DBMS server and host time zones to UTC. This minimizes time zone inconsistencies between applications and data stores. The recommendation is particularly important when replicating between different DBMS types, such as MySQL to Hadoop.
- Use the default time zone settings for Tungsten replicator. Do not change the time zones unless specifically recommended by VMware support.
- If you cannot standardize on UTC at least ensure that time zones are set consistently on all hosts and applications.
F.4. Memory Tuning and Performance

Different areas of Continenut Tungsten 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.
- Tungsten Connector — Memory usage requirements and tuning options.

F.4.1. Understanding Tungsten Replicator Memory Tuning

Replicators are implemented as Java processes, which use two types of memory: stack space, which is allocated per running thread and holds objects that are allocated within individual execution stack frames, and heap memory, which is where objects that persist across individual method calls live. Stack space is rarely a problem for Tungsten as replicators rarely run more than 200 threads and use limited recursion. The Java defaults are almost always sufficient. Heap memory on the other hand runs out if the replicator has too many transactions in memory at once. This results in the dreaded Java OutOfMemory exception, which causes the replicator to stop operating. When this happens you need to look at tuning the replicator memory size.

To understand replicator memory usage, we need to look into how replicators work internally. Replicators use a "pipeline" model of execution that streams transactions through 1 or more concurrently executing stages. As you can 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` [236] 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` [235] parameter at installation time.

- Property `replicator.global.buffer.size` in the `replicator.properties` file. This controls two things, the size of in-memory queues in the replicator as well as the block commit size. If you still have problems after increasing the heap size, try reducing this value. It reduces the number of objects simultaneously stored on the Java heap. A value of 2 is a good setting to try to get around temporary problems. This can be set at installation time as the `--repl-applier-buffer-size` [248] 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` [248] parameter.

- Property `replicator.extractor.dbms.transaction_frag_size` in the `replicator.properties` file. This parameter controls the size of fragments for long transactions. Tungsten automatically breaks up long transactions into fragments. This parameter controls the number of bytes of binlog per transaction fragment. You can try making this value smaller to reduce overall memory usage if many transactions are simultaneously present. Normally however this value has minimal impact.

Finally, it is worth mentioning that the main cause of out-of-memory conditions in replicators is large transactions. In particular, Tungsten cannot fragment individual statements or row changes, so changes to very large column values can also result in OutOfMemory conditions. For now the best approach is to raise memory, as described above, and change your application to avoid such transactions.
F.4.2. Connector Memory Management

The memory model within the Tungsten Connector works as follows:

- Memory consumption consists of the core memory, plus the buffered memory used for each connection.
- Each connection uses the maximum size of an INSERT, UPDATE or SELECT, up to the configured size of the MySQL max_allowed_packet parameter.

For example, with 1000 concurrent connections, and a result or insert size of 1 MB, the memory usage will be 1 GB.

The default setting for the Tungsten Connector memory size is 256 MB. The memory allocation can be increased using tpm and the --conn-java-mem-size [224] option:

For example, during installation:

```
shell> tpm install ... --conn-java-mem-size=1024
```

Or to update using tpm update:

```
shell> tpm update ... --conn-java-mem-size=1024
```
Appendix G. Frequently Asked Questions (FAQ)

The following sections provide the questions and answers to questions often asked by customers and in forums.

G.1. General Questions

G.1.1. How do I update the IP address of one or more hosts in the cluster?

To update the IP address used by one or more hosts in your cluster, you must perform the following steps:

1. If possible, switch the node into SHUNNED mode.
2. Reconfigure the IP address on the machine.
3. Update the hostname lookup, for example, by editing the IP configuration in /etc/hosts.
4. Restart the networking to reconfigure the service.
5. On the node that has changed IP address, run:
   
   ```shell
   tpm update
   ```
   
   The above updates the configuration, but does not restart the individual services, which may still have the old, incorrect, IP address information for the host cached.
6. Restart the node services:
   
   ```shell
   tpm restart
   ```
7. On each other node within the cluster:
   
   a. Update the hostname lookup for the new node, for example, by updating the IP configuration in /etc/hosts.
   b. Update the configuration, using `tpm`:
      
      ```shell
      tpm update
      ```
   c. Restart the services:
      
      ```shell
      tpm restart
      ```

G.1.2. How do I update the password for the replication user in the cluster?

If you need to change the password used by Continuent Tungsten to connect to a dataserver and apply changes, the password can be updated first by changing the information within the your dataserver, and then by updating the configuration using `tpm update`. The new password is not checked until the Tungsten Replicator process is starting. Changing the password and then updating the configuration will keep replication from failing.

1. Within `cctrl` set the maintenance policy mode:
   
   ```cctrl
   set policy maintenance
   ```
2. Within MySQL, update the password for the user, allowing the change to be replicated to the other datasources:
   
   ```mysql
   SET PASSWORD FOR tungsten@'%' = PASSWORD('new_pass');
   ```
3. Follow the directions for `tpm update` to apply the `--datasource-password=new_pass` setting.
4. Set the policy mode in `cctrl` back to `AUTOMATIC`:
   
   ```cctrl
   set policy automatic
   ```

G.1.3. 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, try sending a heartbeat on the master to slaves and checking the latency:

```shell
trepctl heartbeat
```
G.1.4. How do you change the replicator heap size after installation?

You can change the configuration by running the following command from the staging directory:

```
shell> /tools/tpm --host=host1 --java-mem-size=2048
```

G.2. Cloud Deployment and Management

G.2.1. Do we support a 3-node cluster spread across three AWS Availability Zones?

This is a normal deployment pattern for working in AWS to reduce risk. A single cluster works quite well in this topology.

G.2.2. What are the best settings for the Tungsten connector intelligent proxy?

Standard settings work out of the box. Fine tuning can be done by working with the specific customer application during a Proof-Of-Concept or Production roll-out.

G.2.3. How do we use Tungsten to scale DB nodes up/down?

Currently a manual process. New puppet modules to aid this process are being developed, and will be included in the documentation when completed. Here is a link to the relevant procedure Section 2.9.1, “Adding Datasources to an Existing Deployment”.

G.2.4. Do you handle bandwidth/traffic management to the DB servers?

This is not something we have looked at.
Appendix H. Ecosystem Support

In addition to the core utilities provided by Continuent Tungsten, 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 Continuent Tungsten during deployment, monitoring, and management.
- **logrotate** — provides configuration information for users making use of the `logrotate` to manage Continuent Tungsten logs.
- **Cacti** — templates and scripts to enable monitoring through the Cacti environment.
- **Nagios** — templates and scripts to enable monitoring through the Nagios environment.

H.1. Managing Log Files with logrotate

You can manage the logs generated by Continuent Tungsten using `logrotate`.

- **connector.log**

```
/opt/continuent/tungsten/tungsten-connector/log/connector.log {
  notifempty
  daily
  rotate 3
  missingok
  compress
  copytruncate
}
```

- **tmsvc.log**

```
/opt/continuent/tungsten/tungsten-manager/log/tmsvc.log {
  notifempty
  daily
  rotate 3
  missingok
  compress
  copytruncate
}
```

- **trepsvc.log**

```
/opt/continuent/tungsten/tungsten-replicator/log/trepsvc.log {
  notifempty
  daily
  rotate 3
  missingok
  compress
  copytruncate
}
```